

Introduction à l'Intelligence Artificielle



DANIEL RACOCEANU
PROFESSEUR,
SORBONNE UNIVERSITÉ
OCTOBRE 2020
DANIEL.RACOCEANU@SORBONNE-UNIVERSITE.FR



Document confidentiel –
ne peut être reproduit ni diffusé
sans l'accord préalable
de Sorbonne Université.

1

Introduction to Artificial Intelligence



DANIEL RACOCEANU
PROFESOR
SORBONNE UNIVERSITY
OCTOBER 2020
DANIEL.RACOCEANU@SORBONNE-UNIVERSITE.FR



Document confidentiel –
ne peut être reproduit ni diffusé
sans l'accord préalable
de Sorbonne Université.

2

1

Décision dans l'incertain : Processus Markoviens

DANIEL RACOCEANU
PROFESSEUR,
SORBONNE UNIVERSITÉ
OCTOBRE 2020
DANIEL.RACOCEANU@SORBONNE-UNIVERSITE.FR



SORBONNE
UNIVERSITÉ

Document confidentiel –
ne peut être reproduit ni diffusé
sans l'accord préalable
de Sorbonne Université.

3

Decision in uncertain environment: Markov Decision Process (MDP)

DANIEL RACOCEANU
PROFESOR,
SORBONNE UNIVERSITY
OCTOBER 2020
DANIEL.RACOCEANU@SORBONNE-UNIVERSITE.FR



SORBONNE
UNIVERSITÉ

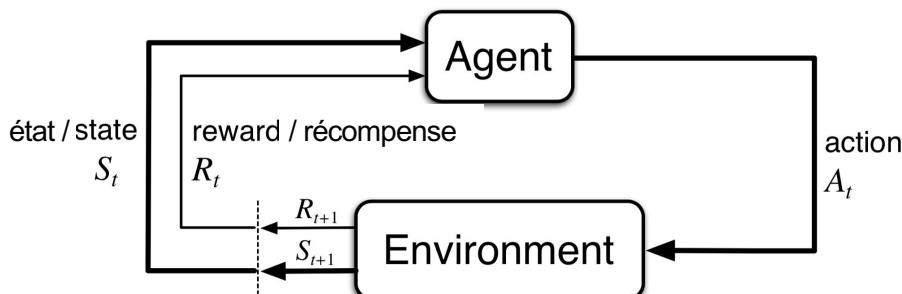
Document confidentiel –
ne peut être reproduit ni diffusé
sans l'accord préalable
de Sorbonne Université.

4

2

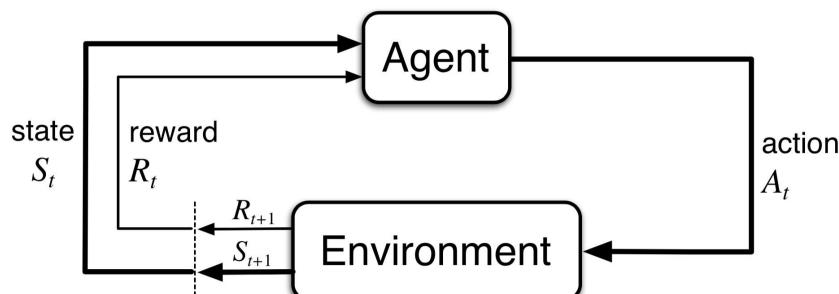
Interface Agent-Environnement

- Agent:
 - apprenti et décideur
- Environnement :
 - ensemble des entités avec lesquelles l'agent interagit, comprenant tout en dehors de l'agent.



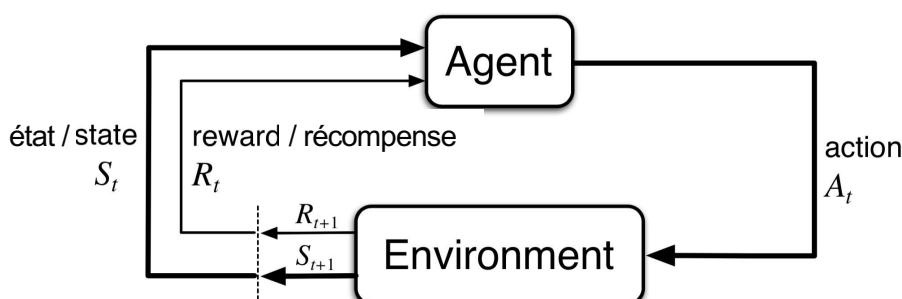
Interface Agent-Environment

- Agent: the learner and decision maker.
- Environment: the thing which the agent interacts with, comprising everything outside the agent.



Interface Agent-Environnement

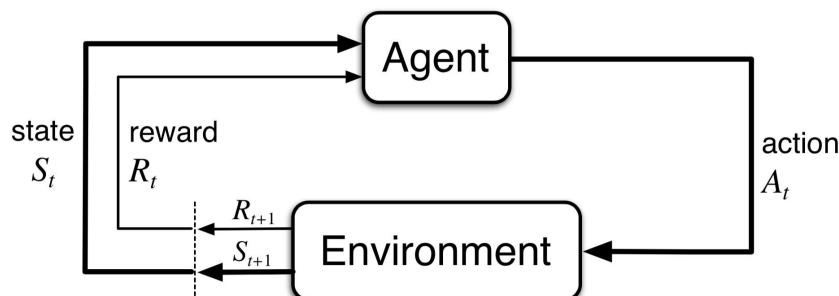
Comment peut-on décider de la frontière agent / environnement ?



7

Interface Agent-Environment

How can we decide the boundary of the agent-environment ?



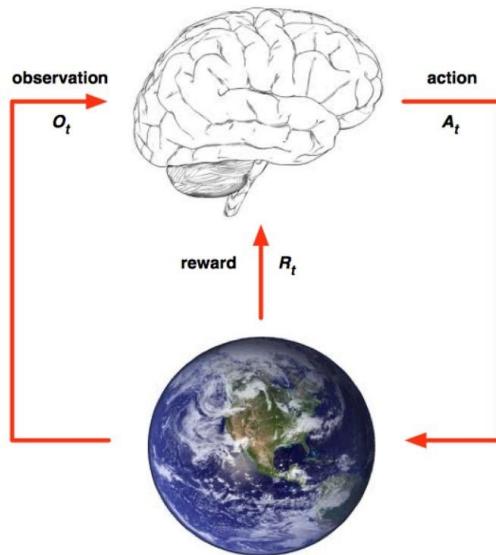
8

Définir les limites entre l'agent et l'environnement

Avant de définir l'ensemble d'états, nous devons définir la frontière entre l'agent et l'environnement.

D'après Richard Sutton :

1. "La frontière agent-environnement représente la limite du contrôle absolu de l'agent, pas de sa connaissance. »
2. "La règle générale que nous suivons est que rien ne peut être changé arbitrairement par l'agent est considéré comme en dehors de celui-ci et donc faisant partie de son environnement."



9

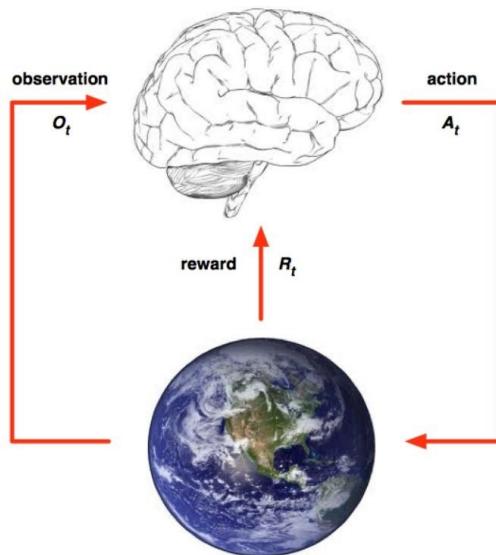
9

Define the boundaries between the agent and the environment

Before defining the set of state, we should define the boundary between agent and environment.

According to Richard Sutton's textbook:

1. "The agent-environment boundary represents the limit of the agent's absolute control, not of its knowledge."
2. "The general rule we follow is that anything cannot be changed arbitrarily by the agent is considered to be outside of it and thus part of its environment."



10

10

Definir les limites agent-environment



11

The Sorbonne Université logo, which consists of a stylized blue 'S' followed by the text 'SORBONNE UNIVERSITÉ'.

11

Système stochastique

- Un **processus stochastique** ou processus aléatoire représente une évolution, discrète ou à temps continu, d'une variable aléatoire.
- Une **variable aléatoire** est une variable dont la valeur est déterminée après un tirage aléatoire.



12

The Sorbonne Université logo, which consists of a stylized blue 'S' followed by the text 'SORBONNE UNIVERSITÉ'.

12

Système stochastique

- Un **processus stochastique** ou processus aléatoire représente une évolution, discrète ou à temps continu, d'une variable aléatoire.
- Une **variable aléatoire** est une variable dont la valeur est déterminée après un tirage aléatoire.
- Système stochastique : un triplet $\Sigma = (S, A, P)$
 - S = ensemble fini d'états
 - A = ensemble fini d'actions
 - $P(s, a, s') =$ probabilité d'aller depuis l'état s vers l'état s' , après exécution de l'action a
 - $\sum_{s' \in S} P(s, a, s') = 1$

13



13

Definir son état / define your state



Chess



Go

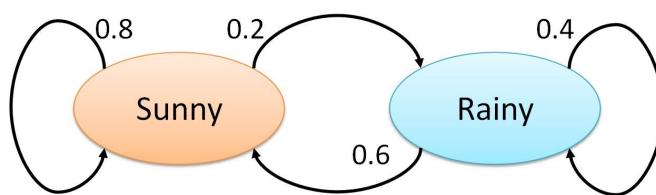
14

14

Processus de décision markovien

En théorie de la décision et de la théorie des probabilités, un **processus de décision markovien** (en anglais Markov Decision Process - MDP) est un modèle stochastique où un agent prend des décisions et où les résultats de ses actions sont aléatoires.

Les MDPs sont utilisés pour étudier des problèmes d'optimisation à l'aide d'algorithmes de **programmation dynamique** ou **d'apprentissage par renforcement**.



15



15

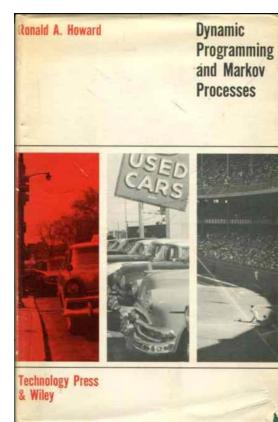
Processus de décision markovien

Les MDPs sont connus depuis les années 1950.

Une grande contribution provient du travail de Ronald A. Howard avec son livre de 1960, *Dynamic Programming and Markov Processes*.

Ils sont utilisés dans de nombreuses disciplines, notamment

- robotique,
- automatisation,
- économie
- industrie manufacturière
- santé



16

16

Processus de décision markovien

Un processus de décision markovien est un processus de contrôle stochastique discret. À chaque étape, le processus est dans un certain état s et l'agent choisit une action a .

La probabilité que le processus arrive à l'état s' est déterminée par l'action choisie.

Plus précisément, elle est décrite par la fonction de transition d'états $T(s, a, s')$.

Donc, l'état s' dépend de l'état actuel s et de l'action a sélectionnée par le décideur. Cependant, pour un s et un a , le prochain état est indépendant des actions et états précédents.

On dit alors que le processus satisfait la **propriété de Markov**.

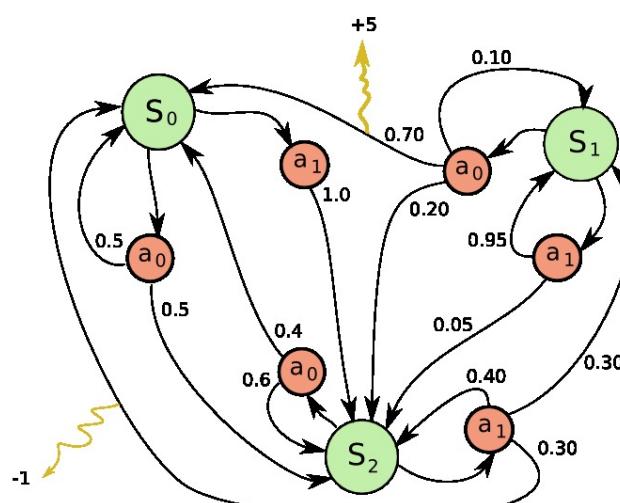
Quand le processus passe de l'état s à l'état s' avec l'action a , l'agent gagne une récompense $R(s, a, s')$.

17



17

Exemple de processus de Décision Markovien à trois états et à deux actions.



18



18

Exemple de processus de Décision Markovien à trois états et à deux actions.

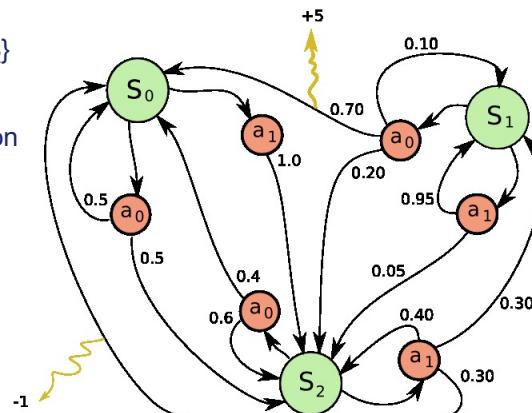
L'exemple donné ci-contre représente un processus de Décision Markovien à trois états distincts $\{s_0, s_1, s_2\}$ représentés en vert.

Depuis chacun des états, on peut effectuer une action de l'ensemble $\{a_0, a_1\}$.

Les nœuds rouges représentent donc une décision possible (le choix d'une action dans un état donné).

Les nombres indiqués sur les flèches sont les probabilités d'effectuer la transition à partir du nœud de décision.

Enfin, les transitions peuvent générer des récompenses (dessinées ici en jaune).



19



19

Exemple de processus de Décision Markovien à trois états et à deux actions.

La matrice de transition associée à l'action a_0 est la suivante :

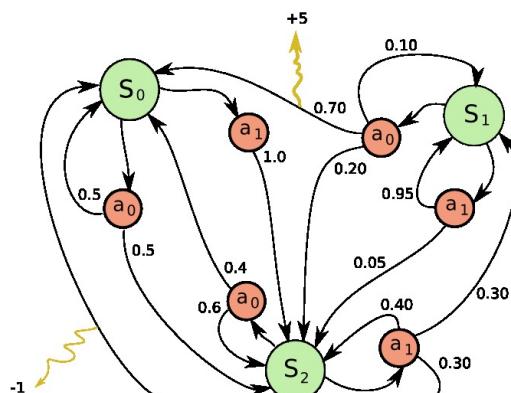
$$\begin{pmatrix} 0.50 & 0 & 0.50 \\ 0.70 & 0.10 & 0.20 \\ 0.40 & 0 & 0.60 \end{pmatrix}$$

La matrice de transition associée à l'action a_1 est la suivante :

$$\begin{pmatrix} 0 & 0 & 1.0 \\ 0 & 0.95 & 0.05 \\ 0.30 & 0.30 & 0.40 \end{pmatrix}$$

En ce qui concerne les récompenses,

- on perçoit une récompense de +5 lorsque l'on passe de l'état s_1 à l'état s_0 en accomplissant l'action a_0
- on perçoit une récompense de -1 (aussi appelée pénalité) lorsque l'on passe de l'état s_2 à l'état s_0 en accomplissant l'action a_1 .



20



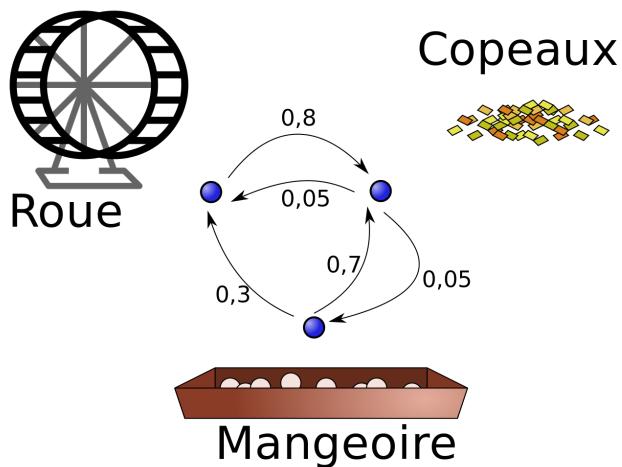
20

Lien MDP – chaînes de Markov

Les MDPs sont une extension des **chaînes de Markov**.

La différence est l'addition des actions choisies par l'agent et des récompenses gagnées par l'agent.

S'il n'y a qu'une seule action à tirer dans chaque état et que les récompenses sont égales, le processus de décision markovien est une chaîne de Markov.



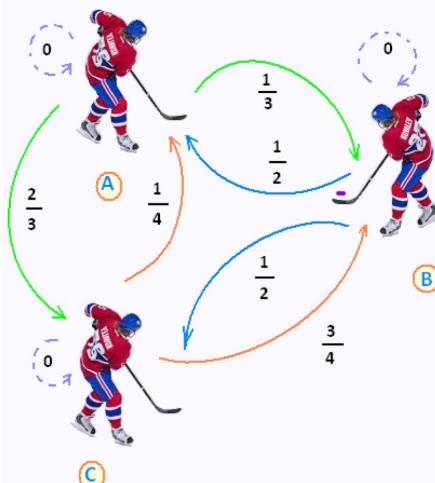
21

Chaines de Markov

Dans une équipe de Hockey, on étudie les passes de la rondelle que font les trois joueurs A, B et C entre eux.

Les probabilités qu'un joueur passe la rondelle à un autre sont représentées sur le graphe ci-dessous.

À partir de ce graphe, on déduit la matrice de transition P correspondante. Si c'est le joueur B qui possède la rondelle au début du jeu, quelle est, par exemple la probabilité que le joueur C la possède après la troisième passe ?



	A	B	C	
Transition de A vers B et C	0	1/3	2/3	A
Transition de B vers A et C	1/2	0	1/2	B
Transition de C vers A et B	1/4	3/4	0	C

matrice de
transition
 G

Example - the gridworld task

http://www.csail.mit.edu/rl/pubs/Chen-ICML09.pdf. Copyright © 2009, Author(s). Licensee Ingenta.



$$R_t = -1 \text{ on all transitions}$$

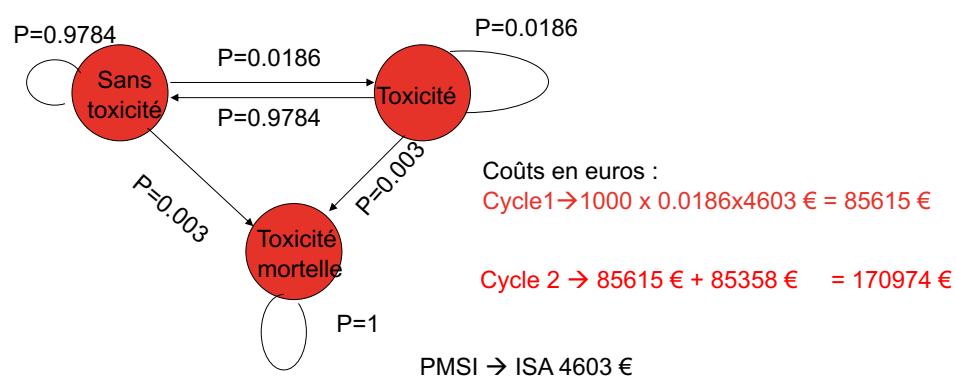
A representation of the gridworld task. Source: Reinforcement Learning: An Introduction (Sutton, R., Barto A.).

23



23

Exemple de simulation de cohorte du coût d'une neutropénie fébrile pour 1000 patients



24

PMSI = « projet de médicalisation des systèmes d'information »
ISA, Indice synthétique d'activité



24

Politique

Une **politique** décrit les choix des actions à jouer par l'agent dans chaque état.

Formellement, il s'agit donc d'une fonction $\pi : S \rightarrow A$ dans le cas d'une **politique déterministe** ou $\pi : S \times A \rightarrow [0,1]$. dans le cas **stochastique**.

On note parfois $\pi(a|s)$ la probabilité de jouer a dans l'état s , i.e. $\mathbb{P}(A_t = a | S_t = s)$, la probabilité de jouer a à l'instant t sachant que l'état à l'instant t est s .

Cette valeur est indépendante de t : on parle de **politique stationnaire**.

Etant donné un MDP et une politique, on obtient une chaîne de Markov avec récompense.

Nous nous plaçons habituellement dans le cas déterministe.

25



25

Critère

L'agent choisit une politique à l'aide de la fonction de **récompense R** .

Notons $r_t = R(s_t, \pi(s_t), s_{t+1})$ la récompense effective obtenue après avoir effectué l'action $\pi(s_t)$ par l'agent qui suit la politique π .

Voici plusieurs **critères d'intérêts** que l'agent peut chercher à maximiser :

- $E \left(\sum_{t=0}^h r_t \right)$: espérance de la somme des récompenses à un horizon fini fixé h ;
- $\liminf_{h \rightarrow +\infty} E \left(\frac{1}{h} \sum_{t=0}^h r_t \right)$ ou $\limsup_{h \rightarrow +\infty} E \left(\frac{1}{h} \sum_{t=0}^h r_t \right)$: récompense moyenne à long terme ;
- $E \left(\sum_{t=0}^{\infty} \gamma^t r_t \right)$: récompense escomptée (ou amortie) à horizon infini où $0 \leq \gamma < 1$.

26



26

Critère

- $E \left(\sum_{t=0}^h r_t \right)$: espérance de la somme des récompenses à un horizon fini fixé h ;
- $\liminf_{h \rightarrow +\infty} E \left(\frac{1}{h} \sum_{t=0}^h r_t \right)$ ou $\limsup_{h \rightarrow +\infty} E \left(\frac{1}{h} \sum_{t=0}^h r_t \right)$: récompense moyenne à long terme ;
- $E \left(\sum_{t=0}^{\infty} \gamma^t r_t \right)$: récompense escomptée (ou amortie) à horizon infini où $0 \leq \gamma < 1$.

Le dernier critère est le plus courant. La valeur de γ permet de définir l'importance que l'on donne au futur.

Quand $\gamma \rightarrow 0$ nous sommes face à un agent «**spéculateur**» qui ne cherche qu'à optimiser son gain immédiat.

À l'opposé si $\gamma \rightarrow 1$, l'agent est «**investisseur**» puisqu'il tient de plus en plus sérieusement compte du futur lointain (si $\gamma = 1$, l'agent tient autant compte du futur lointain que du gain immédiat).

27



27

Fonctions de valeurs

Lorsqu'une politique et un critère sont déterminés, deux fonctions centrales peuvent être définies :

$$V^\pi: S \rightarrow \mathbb{R}$$

est la fonction de **valeur états**; $V^\pi(s)$ représente le gain (selon le critère adopté) engrangé par l'agent s'il démarre à l'état s et applique ensuite la politique π à l'infini.

$$Q^\pi: S \times A \rightarrow \mathbb{R}$$

est la fonction de **valeur états-actions** ; $Q^\pi(s, a)$ représente le gain engrangé par l'agent s'il démarre à l'état s et commence par effectuer l'action a , avant d'appliquer ensuite la politique π à l'infini.

28



28

Équation de Bellman

Les deux fonctions sont intimement liées. On a toujours $V^\pi(s) = Q^\pi(s, \pi(s))$

et, dans le cas du gain amorti à horizon infini, on peut également écrire que:

$$Q^\pi(s, a) = \sum_{s' \in S} [R(s, a, s') + \gamma V^\pi(s')] T(s, a, s').$$

Cette dernière relation montre que la fonction V^π vérifie une relation de récurrence appelée **équation de Bellman**:

$$V^\pi(s) = \sum_{s' \in S} [R(s, \pi(s), s') + \gamma V^\pi(s')] T(s, \pi(s), s').$$

L'équation de Bellman s'écrit comme l'équation linéaire suivante dans la chaîne de Markov avec récompenses "applatie" à partir du processus de décision markovien et la politique π

$$V^\pi = R^\pi + \gamma P^\pi V^\pi$$

où V^π est le vecteur contenant les valeurs pour chaque état, R^π est la matrice des récompenses, P^π est la matrice des probabilités.

29



29

Équations d'optimalité de Bellman

Le but de l'agent est de trouver la politique optimale π^* qui lui permet de maximiser son gain, c'est-à-dire celle qui vérifie, pour $V^{\pi^*}(s) \geq V^\pi(s)$ quelle que soit l'autre politique π .

On peut montrer que la fonction de valeurs optimale V^* vérifie l'**équation d'optimalité de Bellman**:

$$V^*(s) = \max_{a \in A} \sum_{s' \in S} [R(s, \pi(s), s') + \gamma V^*(s')] T(s, a, s').$$

De manière analogue, la fonction Q vérifie elle aussi une équation d'optimalité:

$$Q^*(s, a) = \sum_{s' \in S} [R(s, a, s') + \gamma \max_{a' \in A} Q^*(s', a')] T(s, a, s').$$

30



30

Déterminer la politique optimale: algorithme d'Itération sur la valeur (VI)

La méthode itérative pour les équations d'optimalité de Bellman fournit un premier algorithme, appelé **itération sur la valeur (VI: Value-Iteration)** permettant de déterminer π^* . Il suffit en effet de déterminer V^* avec une précision donnée, et on peut en déduire la politique optimale par:

$$\pi(s) = \arg \max_{a \in A} Q^*(s, a) = \arg \max_{a \in A} \sum_{s' \in S} [R(s, a, s') + \gamma V^*(s')] T(s, a, s').$$

Une difficulté dans cet algorithme est de déterminer la précision avec laquelle calculer V^* de manière à être sûr d'en déduire effectivement la politique optimale.

31



31

Bellman's equation

Iterative policy evaluation

Input π , the policy to be evaluated
 Initialize an array $V(s) = 0$, for all $s \in \mathcal{S}^+$

Repeat

$\Delta \leftarrow 0$

For each $s \in \mathcal{S}$:

$$v \leftarrow V(s) \\ V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r | s, a) [r + \gamma V(s')]$$

$$\Delta \leftarrow \max(\Delta, |v - V(s)|)$$

until $\Delta < \theta$ (a small positive number)

Output $V \approx v_\pi$

32



32

Bellman's equation

Q1

Probability to take action a from state s
following the our policy
(in our case 0.25 since we follow
a random policy and we have 4 actions)

multiplied by the sum of:
the reward r (-1 in our case) plus
the expected value of end state s'
multiplied by a discounting factor
gamma

$$V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) [r + \gamma V(s')]$$

↑ ↑ ↑ ↑
 Value function Sum for all actions Sum for all end states s' and reward r probability to end up in state s' and receive reward r starting in state s and picking action a
 (in our case 1, because actions are deterministic and the reward is always -1)

E

33

Déterminer la politique optimale: algorithme d'Iteration sur la politique (PI)

Un autre algorithme, appelé **itération sur la politique** (PI: Policy-Iteration) essaye d'obtenir la politique optimale sans nécessairement calculer «jusqu'au bout» les valeurs de V^* . L'idée est de partir d'une politique quelconque π_0 , puis d'alterner une phase d'évaluation, dans laquelle la fonction V^{π_n} est déterminée, et une phase d'amélioration, où l'on définit la politique suivante π_{n+1} par:

$$\pi_{n+1}(s) = \arg \max_{a \in A} \sum_{s' \in S} [R(s, a, s') + \gamma V^{\pi_n}(s')] T(s, a, s').$$

Déterminer la politique optimale: algorithme d'Itération sur la politique (PI)

Cet algorithme prend fin lorsqu'aucune évolution de la politique n'est observée : $\pi_{n+1}(s) = \pi_n(s)$ pour tout s .

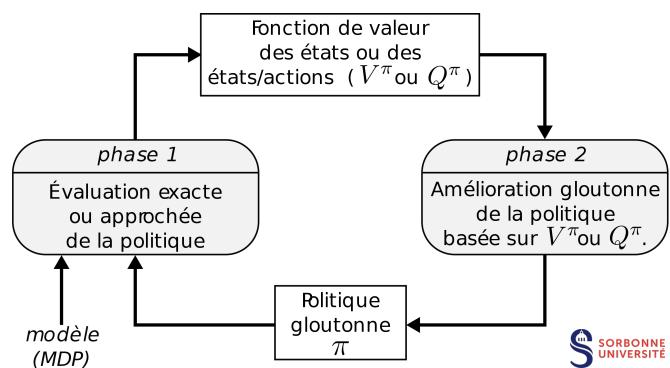
On utilise une méthode itérative pour évaluer V^π , alors se pose la question de savoir à quelle précision s'arrêter.

On peut montrer que même si l'on tronque l'évaluation de V^π , l'algorithme converge tout de même vers l'optimal. À l'extrême, c'est-à-dire lorsqu'une seule itération est utilisée pour évaluer V^π , et après avoir réuni en une seule étape de calcul la phase d'amélioration et la phase d'évaluation, on retombe sur l'algorithme VI.

L'algorithme PI (Policy-Iteration) peut également se formuler dans les termes de la fonction d'états-actions Q plutôt que V .

On voit donc qu'un grand nombre de variantes peuvent être imaginées, mais tournant toutes autour d'un même principe général qui est schématisé à la figure ci-contre.

35



35

Apprentissage par Renforcement Renforcement Learning

DANIEL RACOCEANU
PROFESSEUR,
SORBONNE UNIVERSITÉ
OCTOBRE 2020
DANIEL.RACOCEANU@SORBONNE-UNIVERSITE.FR



Document confidentiel –
ne peut être reproduit ni diffusé
sans l'accord préalable
de Sorbonne Université.

36

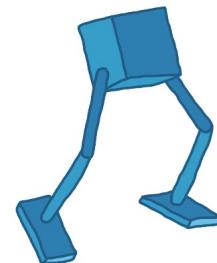
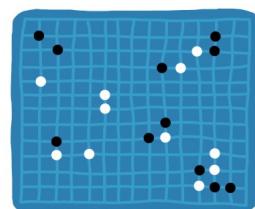
Qu'est-ce l'apprentissage par renforcement ?

Consiste à apprendre ce qu'il faut faire - comment associer des situations à des actions – afin de maximiser un signal de récompense numérique.

L'apprenant n'est pas informé des actions à entreprendre, mais doit plutôt découvrir quelles actions rapportent le plus de récompenses en les essayant.

— Sutton et Barto, *Reinforcement Learning: An Introduction*

37



37

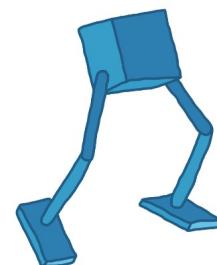
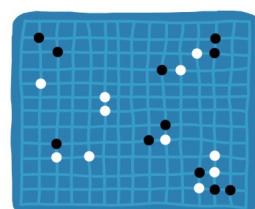
What is Reinforcement Learning?

Learning what to do — how to map situations to actions — so as to maximize a numerical reward signal.

The learner is not told which actions to take, but instead must discover which actions yield the most reward by trying them.

— Sutton and Barto, *Reinforcement Learning: An Introduction*

38



38

Qu'est-ce l'apprentissage par renforcement ?

L'apprentissage par renforcement (RL) a formé avec succès des programmes informatiques pour jouer à des jeux à un niveau supérieur à celui des meilleurs joueurs humains du monde.

Ces programmes trouvent la meilleure action à entreprendre dans les jeux avec de grands espaces d'état et d'action, des informations imparfaites et une incertitude quant à la façon dont les actions à court terme sont bénéfiques à long terme.

39



39

What is Reinforcement Learning?

Reinforcement learning (RL) has successfully trained computer programs to play games at a level higher than the world's best human players.

These programs find the best action to take in games with large state and action spaces, imperfect world information, and uncertainty around how short-term actions pay off in the long run.

40



40

Qu'est-ce l'apprentissage par renforcement ?

Les ingénieurs sont confrontés aux mêmes types de défis lors de la conception de contrôleurs pour des systèmes réels.

L'apprentissage par renforcement peut-il également aider à résoudre des problèmes de contrôle complexes comme faire marcher un robot ou conduire une voiture autonome ?



41



41

What is Reinforcement Learning?

Engineers face the same types of challenges when designing controllers for real systems.

Can reinforcement learning also help solve complex control problems like making a robot walk or driving an autonomous car?



42



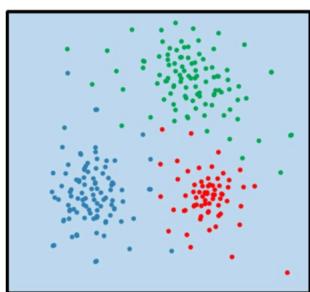
42

Sous-ensemble de l'Apprentissage Automatique

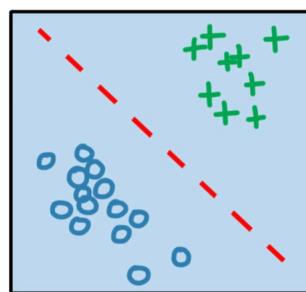
A subset of Machine Learning

machine learning

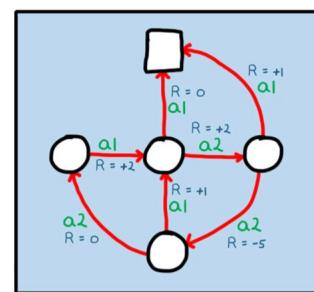
unsupervised
learning



supervised
learning



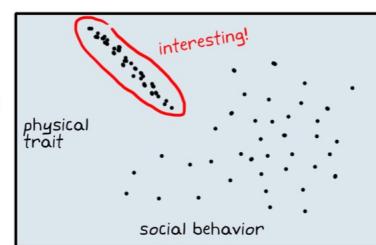
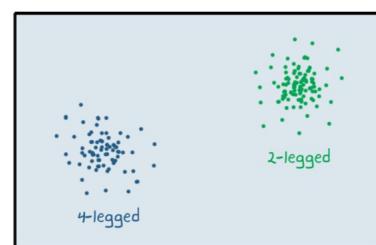
reinforcement
learning



Apprentissage non-supervisé

Utilisé pour trouver des paramètres ou structures cachées dans les bases de données non classifiées / non-labelisées.

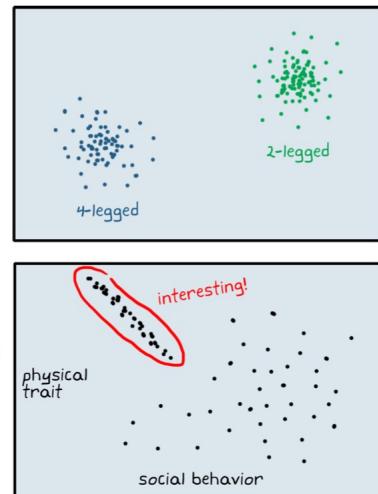
animal dataset (unlabeled)					
weight	height	num. of legs	communal living	domesticatable	-
1.3	4.5	2	yes	no	-
11.2	1.9	4	yes	yes	-
9.5	2.2	4	no	no	-
15	5.1	4	yes	yes	-
1.3	0.8	6	no	no	-
-	-	-	-	-	-



Unsupervised Learning

Used to find patterns or hidden structures in datasets that have not been categorized or labeled.

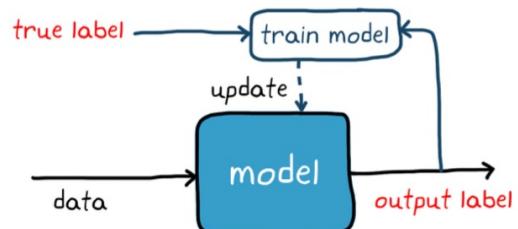
animal dataset (unlabeled)					
weight	height	num. of legs	communal living	domesticatable	-
1.3	4.5	2	yes	no	-
11.2	1.8	4	yes	yes	-
0.5	2.2	4	no	no	-
15	5.1	4	yes	yes	-
1.3	0.8	6	no	no	-
-	-	-	-	-	-



Apprentissage supervisé Supervised Learning

Entraînez l'ordinateur à appliquer une étiquette à une entrée donnée.
Train the computer to apply a label to a given input.

species	animal dataset (labeled)				
	weight	height	num. of legs	communal living	domesticatable
rat	1.3	11	4	yes	yes
robin	1.2	0.8	4	no	no
elephant	40.5	12.2	4	yes	no
rabbit	2.5	2.1	4	yes	yes
spider	0.1	0.2	8	no	no
-	-	-	-	-	-



Apprentissage par Renforcement

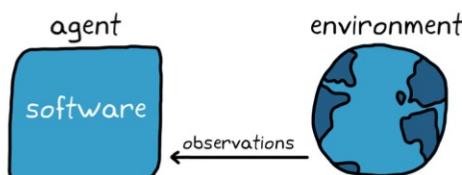
Contrairement aux deux autres cadres d'apprentissage, qui fonctionnent à l'aide d'un ensemble de **données statique**, l'AR fonctionne avec des données provenant d'un **environnement dynamique**. L'objectif n'est pas de regrouper des données ou d'étiqueter des données, mais de trouver la meilleure séquence d'actions qui générera le résultat optimal.

L'apprentissage par renforcement résout ce problème en permettant à un logiciel appelé agent, d'explorer, d'interagir avec et d'apprendre de l'environnement.

47

1

The agent is able to observe the current state of the environment.



47

Reinforcement Learning

Unlike the other two learning frameworks, which operate using a **static dataset**, RL works with data from a **dynamic environment**.

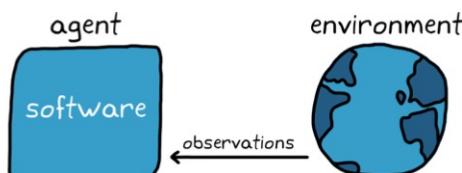
The goal is not to cluster data or label data, but to find the **best sequence of actions that will generate the optimal outcome**.

The way reinforcement learning solves this problem is by allowing a piece of software called an agent to explore, interact with, and learn from the environment.

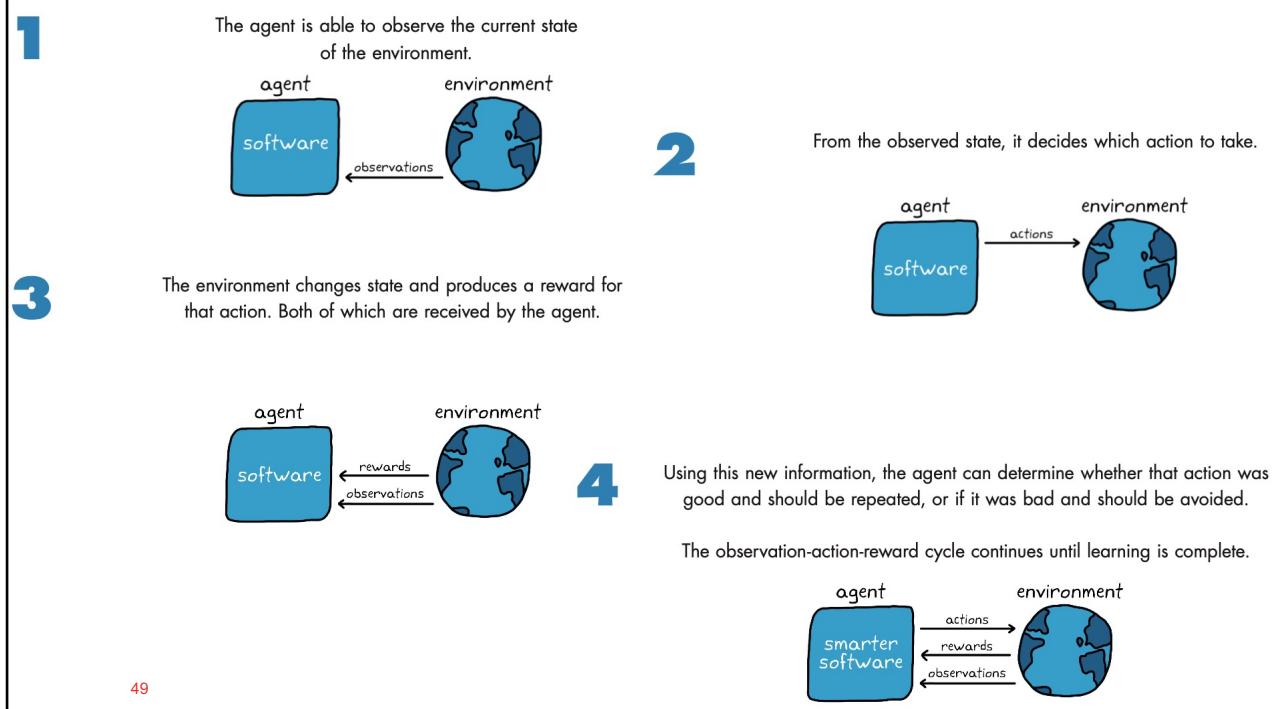
48

1

The agent is able to observe the current state of the environment.



48



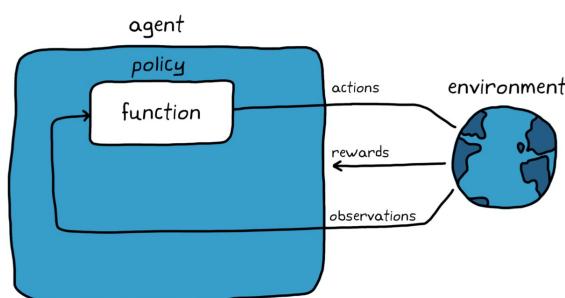
49

Apprentissage par Renforcement (AR)

L'agent possède une fonction qui prend en compte les observations d'état (les entrées) et les mappe à des actions (les sorties).

Il s'agit de la fonction unique qui remplacera tous les sous-composants individuels de votre système de contrôle.

Dans la nomenclature AR, cette fonction est appelée la politique. Compte tenu d'un ensemble d'observations, la politique décide de l'action à entreprendre.



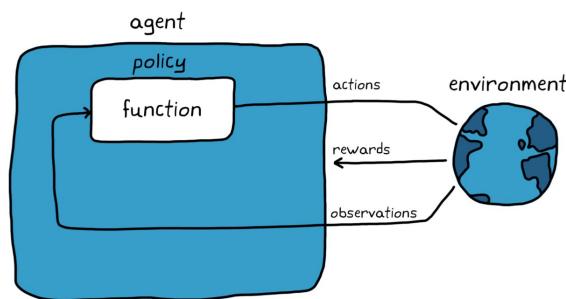
50

Reinforcement Learning (RL)

Within the agent, there is a function that takes in state observations (the inputs) and maps them to actions (the outputs). T

This is the single function that will take the place of all of the individual subcomponents of your control system.

In the RL nomenclature, this function is called the policy. Given a set of observations, the policy decides which action to take.



51

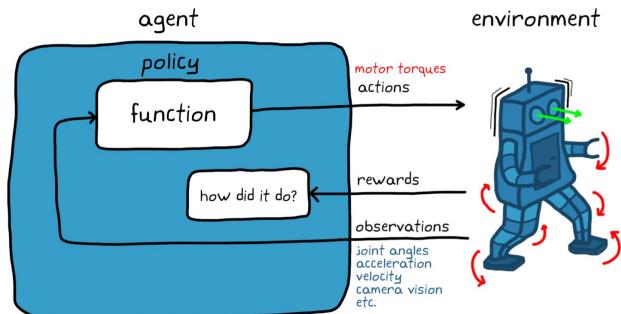


51

Apprentissage par Renforcement

Dans l'exemple du robot marcheur, les observations seraient l'angle de chaque articulation, l'accélération et la vitesse angulaire du tronc du robot et les milliers de pixels du capteur de vision.

La politique prendrait en compte toutes ces observations et produirait les commandes motrices qui déplaceront les bras et les jambes du robot.



52

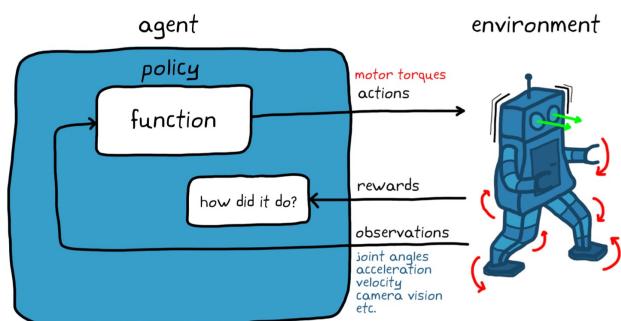


52

Reinforcement Learning

In the walking robot example, the observations would be the angle of every joint, the acceleration and angular velocity of the robot trunk, and the thousands of pixels from the vision sensor.

The policy would take in all of these observations and output the motor commands that will move the robot's arms and legs.

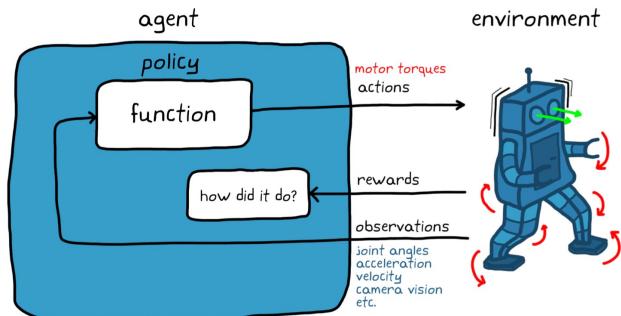


53

53

Apprentissage par Renforcement

L'environnement générerait alors une récompense indiquant à l'agent à quel point la combinaison très spécifique de commandes d'actionneur a fonctionné. Si le robot reste debout et continue de marcher, la récompense sera plus élevée que si le robot tombait au sol.



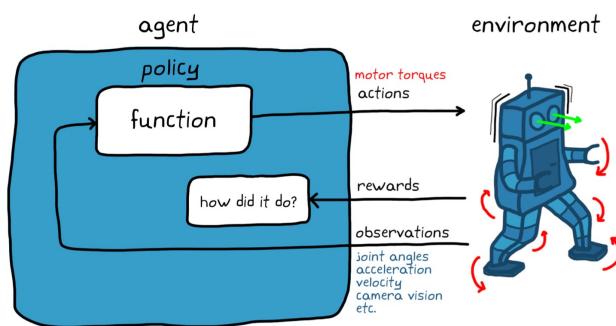
54

54

Reinforcement Learning

The environment would then generate a reward telling the agent how well the very specific combination of actuator commands did.

If the robot stays upright and continues walking, the reward will be higher than if the robot falls to the ground.



55

55

Apprendre la politique optimale

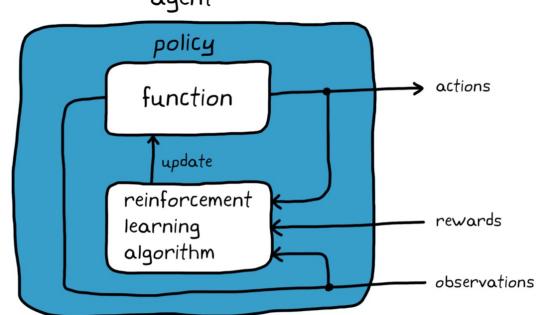
Si vous pouviez concevoir une politique parfaite qui commanderait correctement les bons actionneurs pour chaque état observé, alors votre travail serait fait.

Bien sûr, ce serait difficile à faire dans la plupart des situations. Même si vous trouvez la stratégie parfaite, **l'environnement pourrait changer au fil du temps**, de sorte qu'un mappage statique ne serait plus optimal.

Cela nous amène à l'algorithme d'apprentissage par renforcement :

Il modifie la politique en fonction des actions entreprises, des observations de l'environnement et du montant de la récompense collectée.

L'objectif de l'agent est **d'apprendre la meilleure politique au fur et à mesure qu'il interagit avec l'environnement** afin que, quel que soit l'état, il entreprenne toujours l'action la plus optimale - celle qui produira le plus de récompenses.



56

28

Learning the Optimal Policy

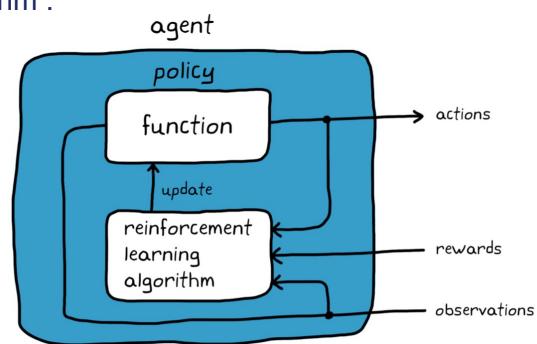
If you were able to design a perfect policy that would correctly command the right actuators for every observed state, then your job would be done.

Of course, that would be difficult to do in most situations. Even if you did find the perfect policy, **the environment might change over time**, so a static mapping would no longer be optimal.

This brings us to the reinforcement learning algorithm :

It changes the policy based on the actions taken, the observations from the environment, and the amount of reward collected.

The goal of the agent is to **learn the best policy as it interacts with the environment** so that, given any state, it will always take the most optimal action — the one that will produce the most reward in the long run.



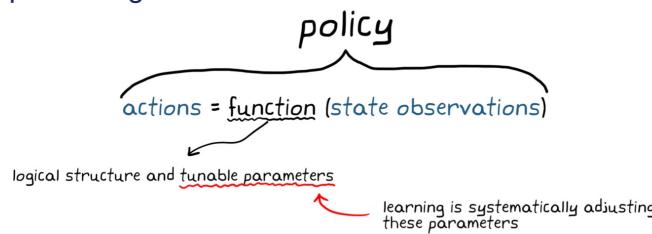
57

Que signifie “apprendre” ?

Pour comprendre ce que cela signifie pour une machine d'apprendre, réfléchissez à ce qu'est réellement **une politique** : une fonction composée d'une logique et de paramètres ajustables.

Étant donnée une politique, il existe un ensemble de paramètres qui produiront une politique optimale - une cartographie des états aux actions qui produisent la récompense la plus à long terme. L'apprentissage est le terme donné au **processus d'ajustement systématique de ces paramètres pour converger vers la politique optimale**. De cette façon, vous pouvez vous concentrer sur la mise en place d'une structure de stratégie adéquate sans régler manuellement la fonction pour obtenir les bons paramètres.

Vous pouvez laisser l'ordinateur apprendre les paramètres par lui-même grâce à un processus d'apprentissage.



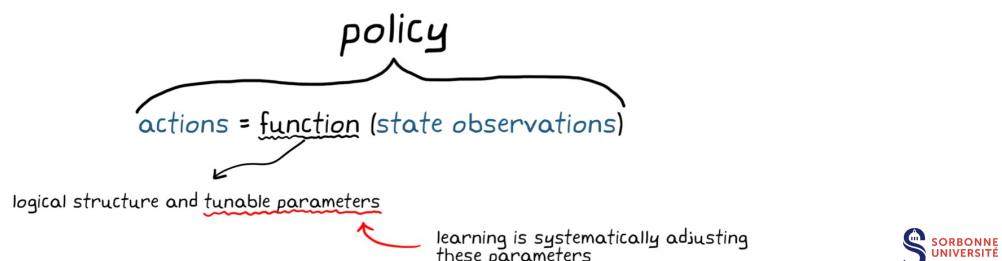
58

What Does It Mean to Learn?

To understand what it means for a machine to learn, think about what a policy actually is: a function made up of logic and tunable parameters.

Given a policy structure, there is a set of parameters that will produce an optimal policy – i.e. a mapping of states to actions that produces the most long-term reward. Learning is the term given to the process of **systematically adjusting those parameters to converge on the optimal policy**. In this way, you can focus on setting up an adequate policy structure without manually tuning the function to get the right parameters.

You can let the computer learn the parameters on its own through a learning process.



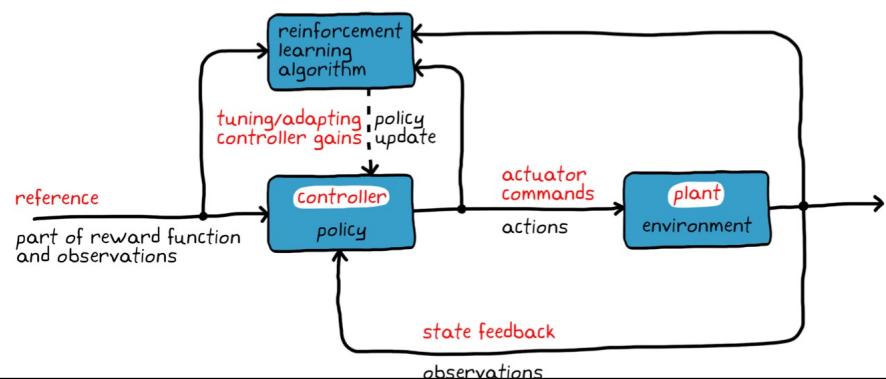
59

Similitude entre l'A/R et l'automatique

L'objectif de l'apprentissage par renforcement est similaire au problème du contrôle : c'est juste une approche qui utilise des termes différents pour représenter les mêmes concepts.

Avec les deux méthodes, vous souhaitez déterminer les entrées correctes dans un système qui généreront le comportement système souhaité. Vous essayez de comprendre comment concevoir la **politique** (ou le **contrôleur**) qui mappe l'état observé de l'environnement (système industriel) aux meilleures **actions** (**commandes de l'actionneur**).

Le signal de retour d'état correspond aux observations de l'environnement, et le signal de référence est intégré à la fois à la fonction de récompense et aux observations de l'environnement.



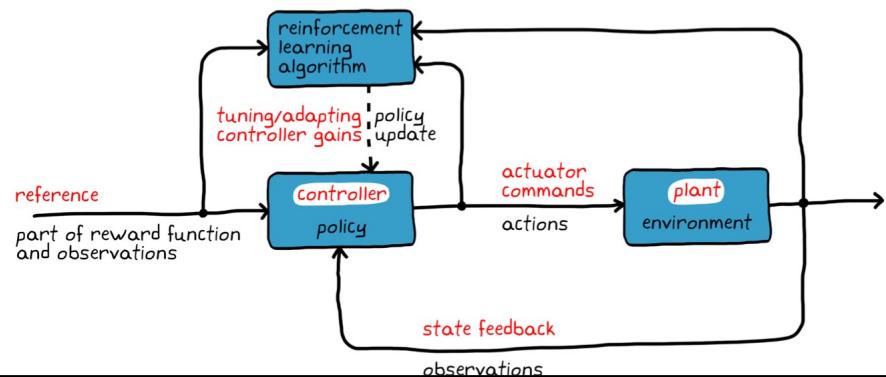
60

Similarity of RL with Traditional Controls

The goal of reinforcement learning is similar to the control problem : it's just using different terms to represent the same concepts.

With both methods, you want to determine the correct inputs into a system that will generate the desired system behavior. You are trying to figure out how to design the **policy** (or the **controller**) that maps the observed state of the environment (or the plant) to the best **actions** (the **actuator commands**).

The state feedback signal is the observations from the environment, and the reference signal is built into both the reward function and the environment observations.



61

Workflow d'apprentissage par renforcement

Reinforcement Learning Workflow Overview

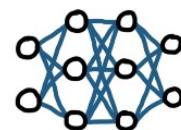
environment



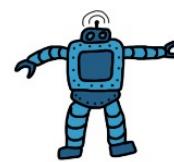
reward



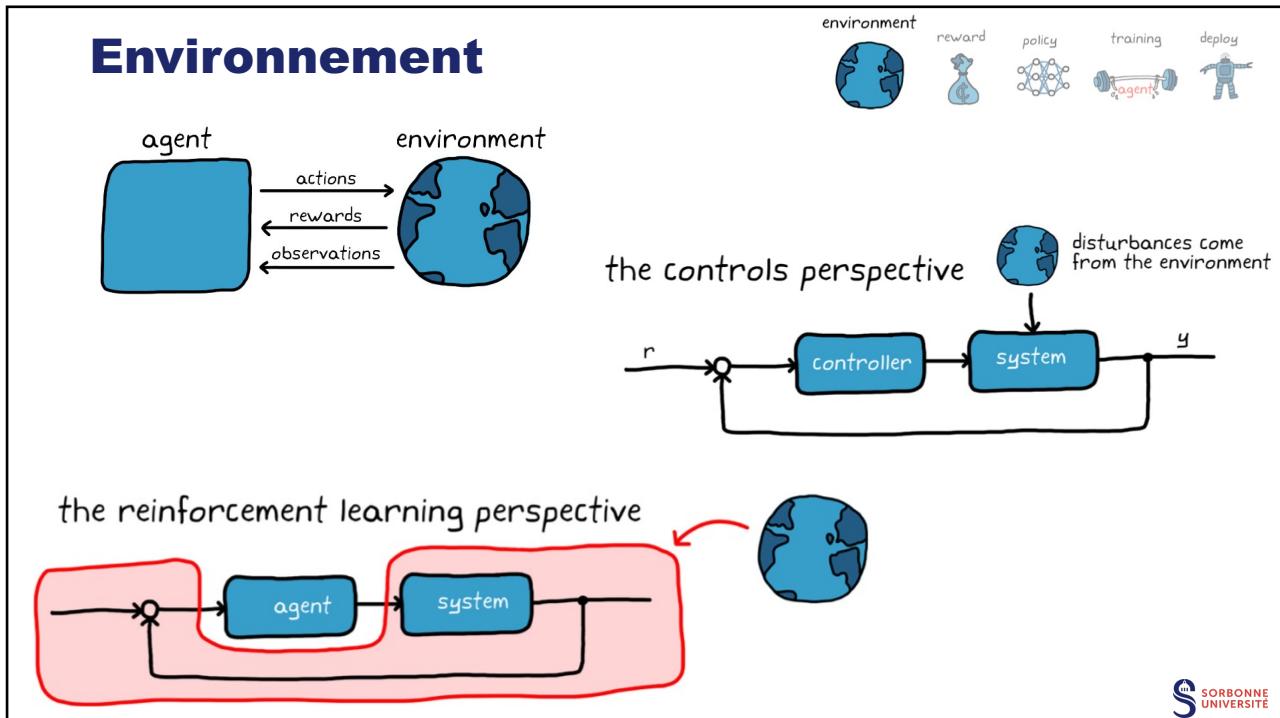
policy



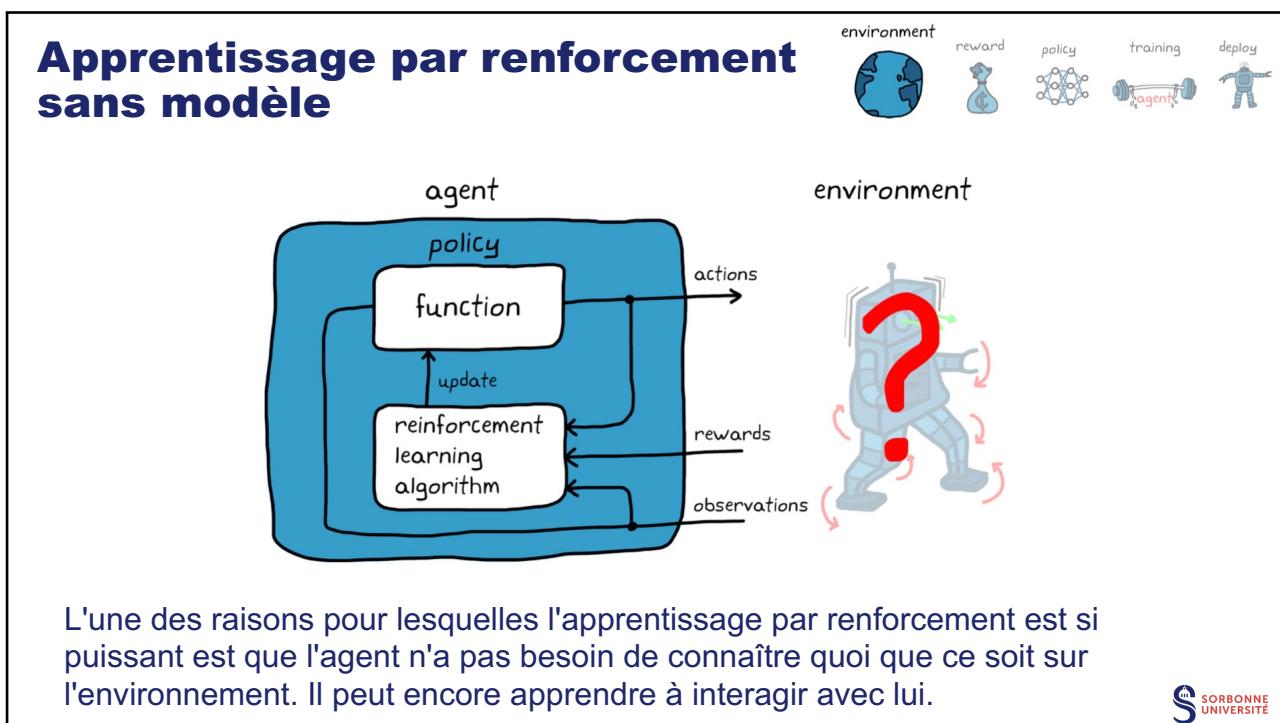
deploy



62

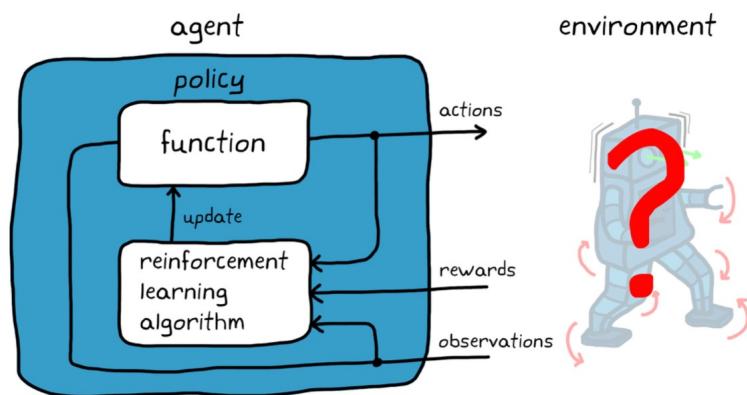
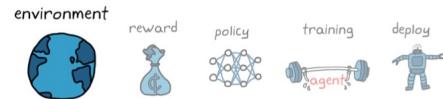


63



64

Model-Free Reinforcement



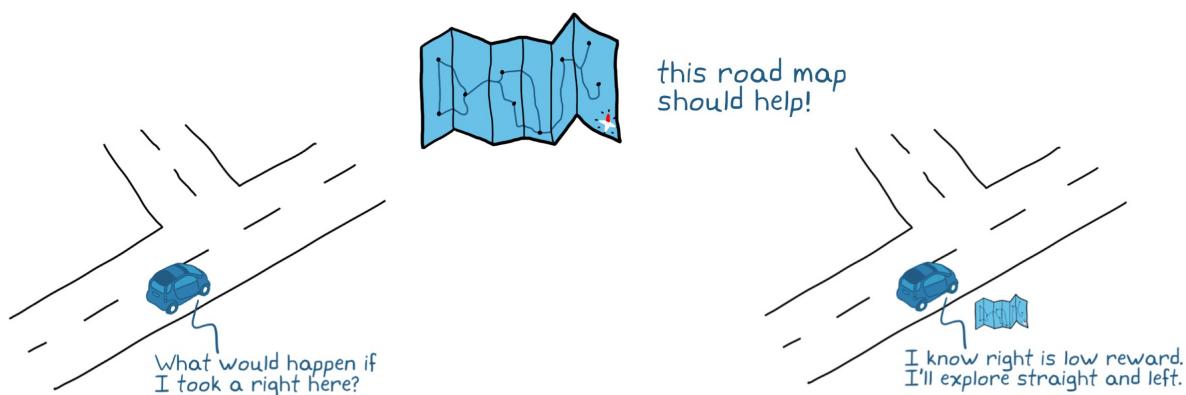
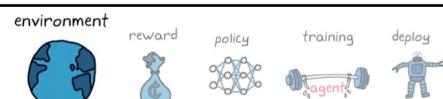
One reason reinforcement learning is so powerful is that the agent does not need to know anything about the environment. It can still learn how to interact with it.

65



65

Apprentissage par renforcement basé sur un modèle

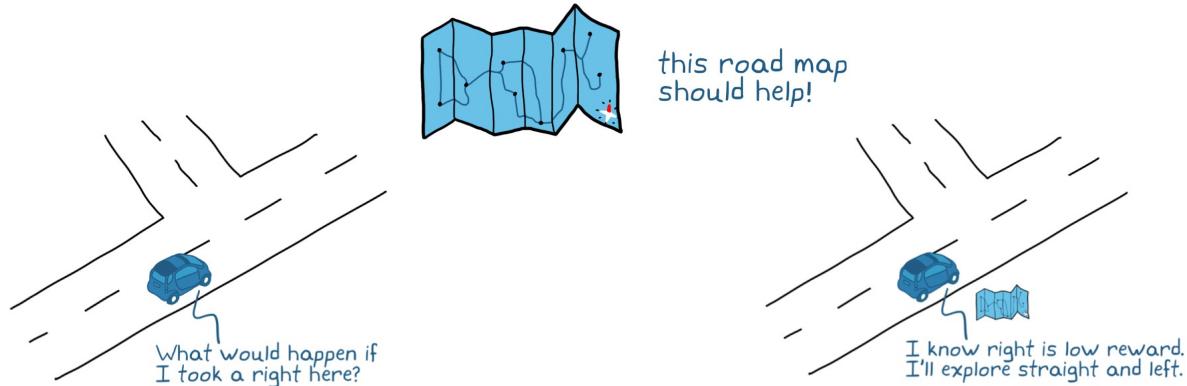
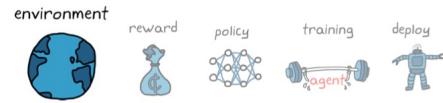


À l'aide d'un modèle, l'agent peut explorer des parties de l'environnement sans avoir à effectuer physiquement cette action. Un modèle peut compléter le processus d'apprentissage en évitant les domaines connus pour être mauvais et en explorant le reste.



66

Model-Based Reinforcement Learning

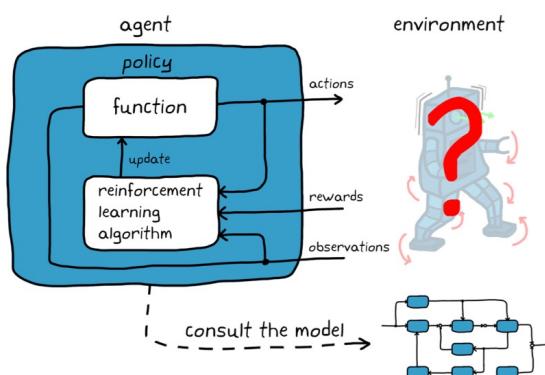
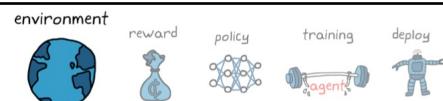


Using a model, the agent can explore parts of the environment without having to physically take that action. A model can complement the learning process by avoiding areas⁶ that are known to be bad and exploring the rest.



67

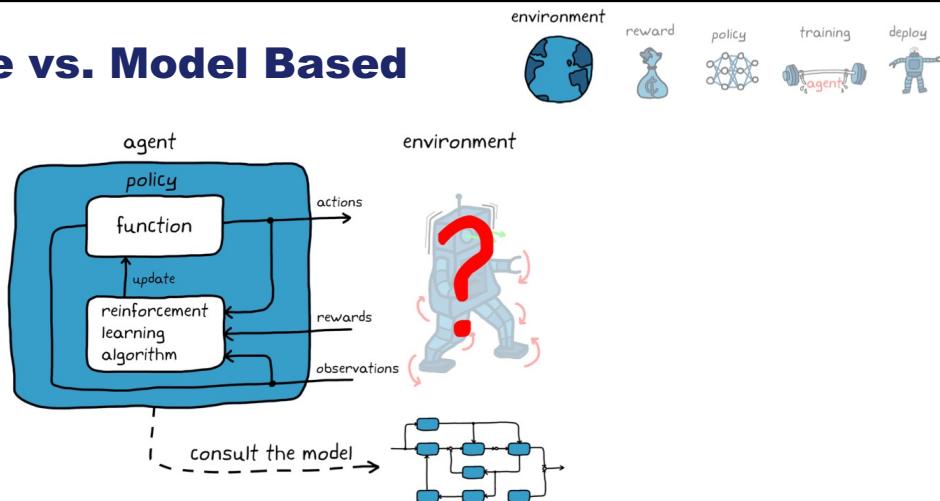
Sans vs. avec modèle



L'apprentissage par renforcement basé sur un modèle peut réduire le temps nécessaire à l'apprentissage d'une politique optimale, car le modèle permet d'éloigner l'agent des zones de l'espace d'état dont vous savez qu'elles ont de faibles récompenses. Si vous comprenez les bases de l'apprentissage par renforcement sans modèle, il est plus intuitif de continuer avec le RL basé sur un modèle.

68

Model Free vs. Model Based

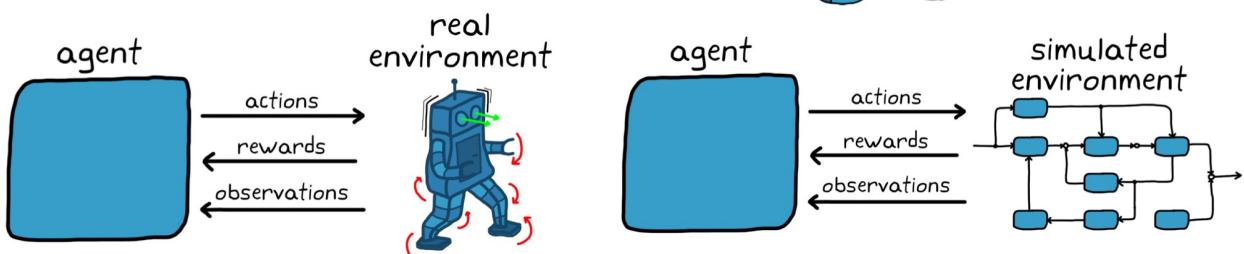


Model-based reinforcement learning can lower the time to learn an optimal policy because you can use the model to guide the agent away from areas of the state space that you know having low rewards.

If you understand the basics of reinforcement learning without a model, then continuing on to model-based RL is more intuitive.

69

Environnements réels ou simulés



REEL

Précision : Rien ne représente l'environnement plus complètement que l'environnement réel.

Simplicité : Il n'est pas nécessaire de perdre du temps à créer et à valider un modèle.

Nécessaire : Il peut être nécessaire de s'entraîner avec l'environnement réel s'il est en constante évolution ou difficile à modéliser avec précision.

70

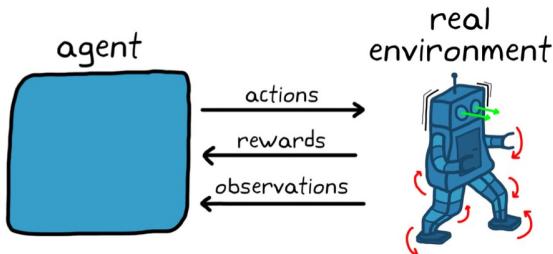
SIMULE

Vitesse : les simulations peuvent s'exécuter plus rapidement qu'en temps réel ou être parallélisées, accélérant ainsi un processus d'apprentissage lent.

Conditions simulées : Il est plus facile de modéliser des situations qui seraient difficiles à tester.

Sécurité : Il n'y a aucun risque d'endommager le matériel.

Real vs. Simulated Environments

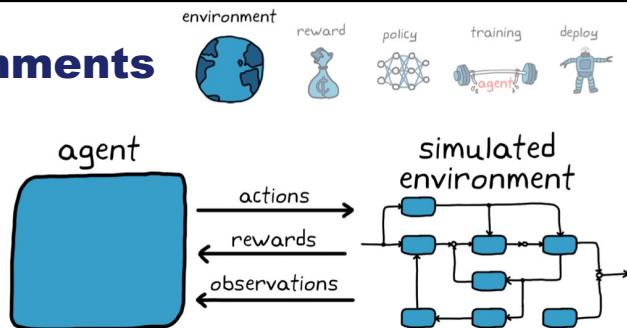


REAL

Accuracy: Nothing represents the environment more completely than the real environment.

Simplicity: There is no need to spend the time creating and validating a model.

Necessary: It might be necessary to train with the real environment if it is constantly changing or difficult to model accurately.



SIMULATED

Speed: Simulations can run faster than real time or be parallelized, speeding up a slow learning process.

Simulated conditions: It is easier to model situations that would be difficult to test.

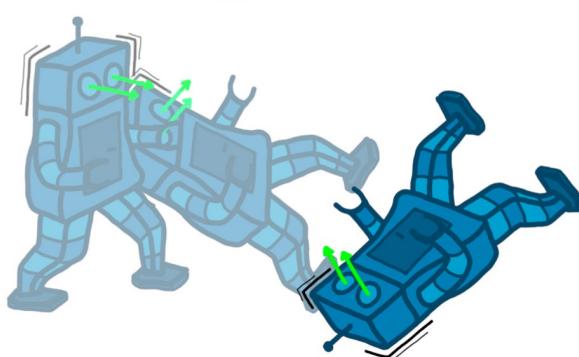
Safety: There is no risk of damage to hardware.

71

Environnements réels ou simulés



Vous pouvez laisser un agent apprendre à équilibrer un pendule inversé en l'exécutant avec une configuration de pendule physique. Cela pourrait être une bonne solution car il est probablement difficile pour le matériel de s'endommager ou d'endommager les autres. Étant donné que les espaces d'état et d'action sont relativement petits, l'entraînement ne prendra probablement pas trop de temps.



Avec le robot marcheur, si la politique n'est pas suffisamment optimale lorsque vous démarrez l'entraînement, le robot va faire beaucoup de chutes avant même d'apprendre à bouger ses jambes, et encore moins à marcher. Non seulement cela pourrait endommager le matériel, mais devoir ramasser le robot à chaque fois prendrait énormément de temps.

72

Real vs. Simulated Environments

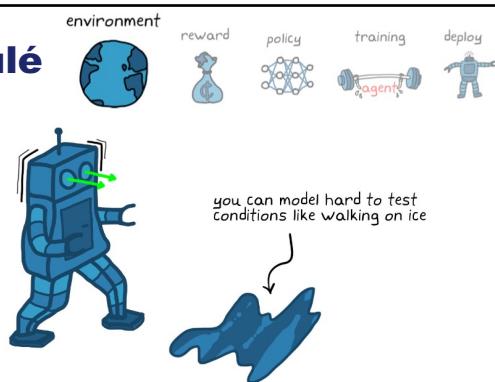
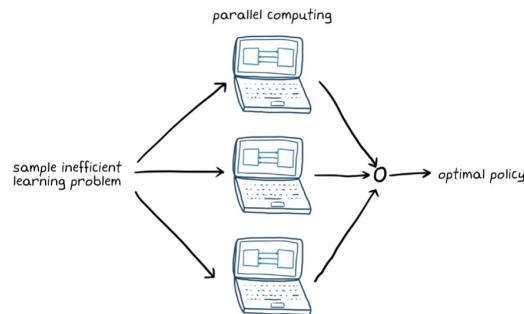


You could let an agent learn how to balance an inverted pendulum by running it with a physical pendulum setup. This might be a good solution since it's probably **hard for the hardware to damage itself or others**. Since the state and action spaces are relatively small, it probably won't take too long to train.

With the walking robot, if the policy is not sufficiently optimal when you start training, the robot is going to do a lot of **falling and flailing** before it even learns how to move its legs, let alone how to walk. Not only could this damage the hardware, but having to pick the robot up each time would be extremely time consuming.

73

Avantages d'un environnement simulé

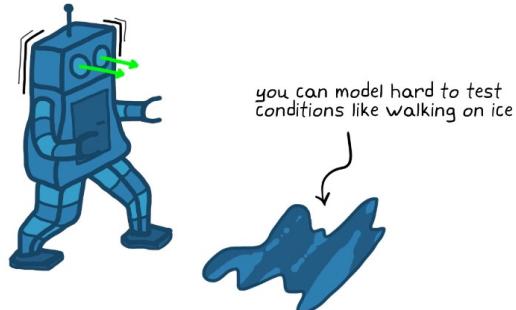
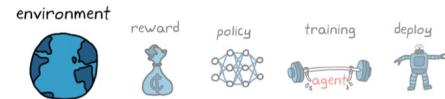
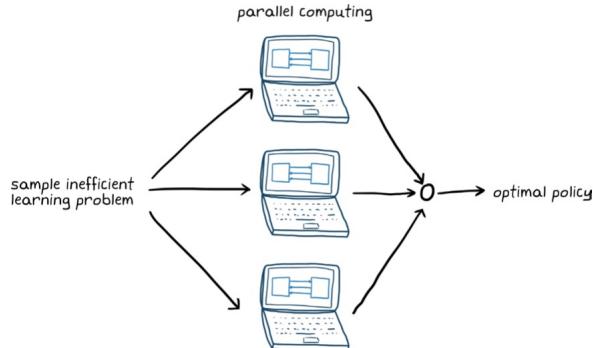


L'apprentissage est un processus qui nécessite de nombreux échantillons : essais, erreurs et corrections. Il est très inefficace dans ce sens car cela peut prendre des milliers / millions d'épisodes pour converger vers une solution optimale. **Un modèle de l'environnement peut s'exécuter plus rapidement qu'en temps réel et vous pouvez lancer de nombreuses simulations à exécuter en parallèle.** Ces deux approches peuvent accélérer le processus d'apprentissage.

Vous avez beaucoup plus de contrôle sur la simulation des conditions que d'exposez votre agent au monde réel. Par exemple, votre robot devra peut-être être capable de marcher sur un **nombre illimité de surfaces différentes**. Simuler la marche sur une surface à faible frottement comme la glace est beaucoup plus simple que de tester sur de la glace réelle. Il est possible de créer un **mieux environnement d'entraînement avec la simulation**.

74

Benefits of a Simulated Environment

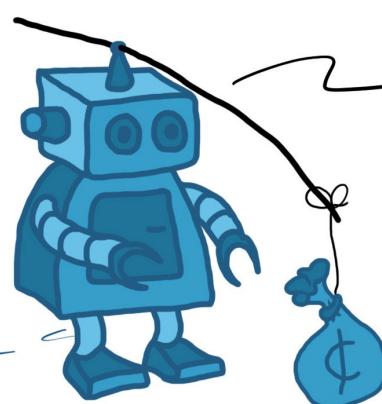
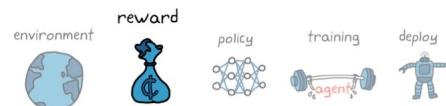


Learning is a process that requires lots of samples: trials, errors, and corrections. It is very inefficient in this sense because it can take thousands or millions of episodes to converge on an optimal solution. **A model of the environment may run faster than real time**, and you can spin up lots of simulations to **run in parallel**. Both of these approaches can speed up the learning process.

You have a lot more control over simulating conditions than you do exposing your agent to them in the real world. For example, your robot may have to be capable of walking on any number of **different surfaces**. Simulating walking on a low-friction surface like ice is much simpler than testing on actual ice. It's possible to create a **better training environment with simulation**.

75

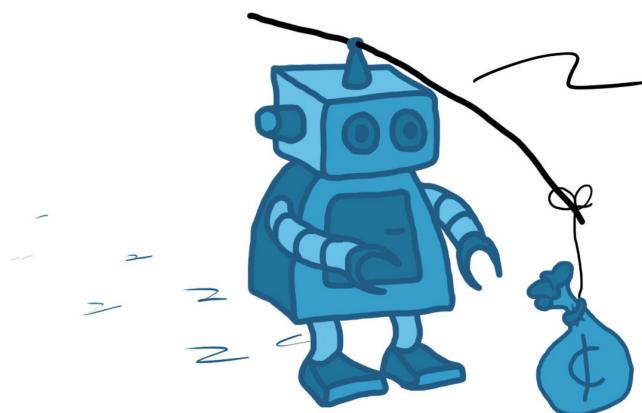
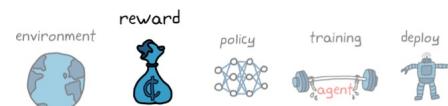
Les Revenu



Avec l'environnement défini, on doit concevoir une fonction de récompense afin que l'algorithme d'apprentissage « comprenne » quand la politique s'améliore et converge vers le résultat attendu.

76

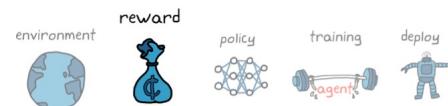
The Reward



With the environment set, the next step is to think about what you want your agent to do and how you'll reward it for doing what you want. This requires crafting a reward function so that the learning algorithm "understands" when the policy is getting better and ultimately converges on the result you're looking for.

77

What is the Reward?



reward = function (state, action)

scalar representing "goodness"

$$\text{LQR cost function, } J = \int_0^{\infty} (x^T Q x + u^T R u) dt$$

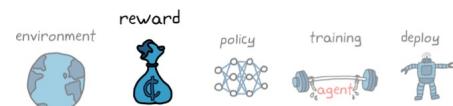
↑ ↓ ↓
performance quadratic effort

Linear Quadratic Regulator (LQR)



78

Sparse Rewards



$$\text{reward} = \begin{cases} 1 & \text{for state} = 10 \text{ meters} \\ 0 & \text{for state} \approx 10 \text{ meters} \end{cases}$$

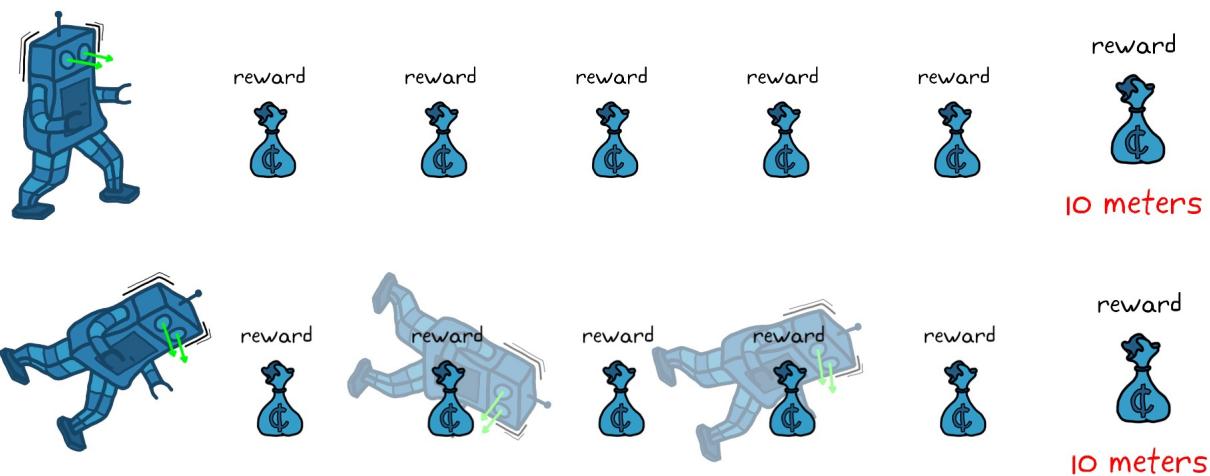
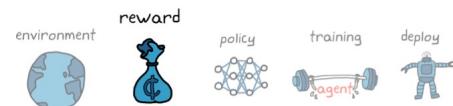


79

SORBONNE
UNIVERSITÉ

79

Reward Shaping



80

SORBONNE
UNIVERSITÉ

80

Domain-Specific Knowledge

The diagram illustrates a robot walking on a ramp. A coordinate system is shown with axes x and z . The initial height of the ramp is labeled "initial height". The robot's trunk height is indicated by a vertical line. The reward function equation is:

$$r_t = v_x - 3y^2 - 50\hat{z}^2 + 25 \frac{T_s}{T_f} - 0.02 \sum_i u_{t-1}^i$$

Annotations explain the terms:

- v_x : forward velocity
- $3y^2$: don't stray from path
- $50\hat{z}^2$: keep trunk high
- $25 \frac{T_s}{T_f}$: walk as long as possible
- $0.02 \sum_i u_{t-1}^i$: minimize actuator effort

81

81

Exploration vs. Exploitation

The diagram shows a sequence of states s_1, s_2, s_3 connected by arrows. Action a_1 leads from s_2 to s_1 , while action a_2 leads from s_1 to s_3 . The current state is s_1 .

If action a_1 is taken, then the agent receives information about state s_2 and nothing about state s_3 .

Un aspect critique de l'apprentissage par renforcement est le compromis entre l'exploration et l'exploitation pendant qu'un agent interagit avec un environnement. La raison pour laquelle cette décision vient avec l'apprentissage par renforcement est que l'apprentissage se fait en ligne. Au lieu de travailler à partir d'un ensemble de données statique, les actions de l'agent déterminent quelles données sont renvoyées de l'environnement. Les choix que fait l'agent déterminent les informations qu'il reçoit et, par conséquent, les informations à partir desquelles il peut apprendre.

L'idée est la suivante : l'agent doit-il exploiter l'environnement en choisissant les actions qui collectent le plus de récompenses qu'il connaît déjà, ou doit-il choisir des actions qui explorent des parties de l'environnement encore inconnues ?

82

current state s_1

if action a_1 is taken, then the agent receives information about state s_2 and nothing about state s_3

exploit

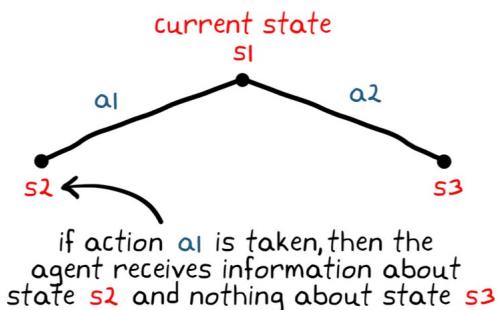
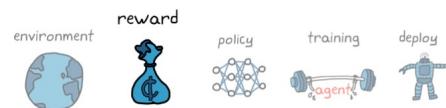
explore

reward: +!

reward: ??

82

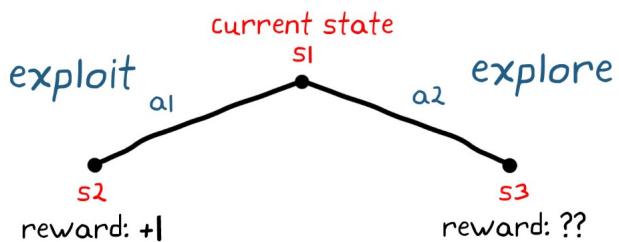
Exploration vs. Exploitation



The idea is this: Should the agent exploit the environment by choosing the actions that collect the most rewards that it already knows about, or should it choose actions that explore parts of the environment that are still unknown?

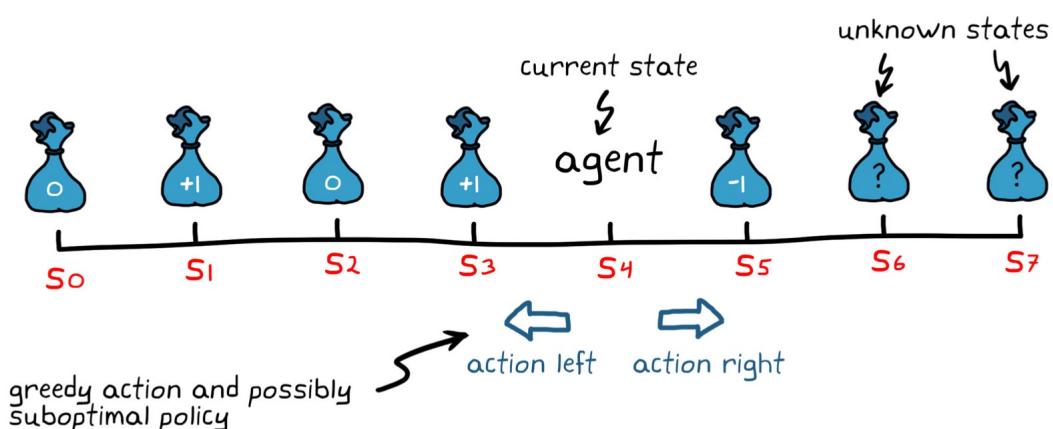
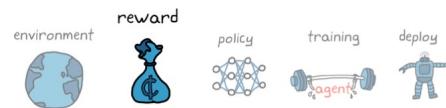
83

A critical aspect of reinforcement learning is the tradeoff between **exploration** and **exploitation** while an agent interacts with an environment. The reason this decision comes up with reinforcement learning is that **learning is done online**. Instead of working from a static dataset, the agent's actions determine which data is returned from the environment. The choices the agent makes determine the information it receives and, therefore, the information from which it can learn.



83

The Problem with Pure Exploitation

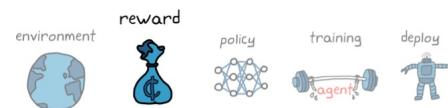


84



84

The Problem with Pure Exploration



what if I take
a hard right?
that's not a
good idea!

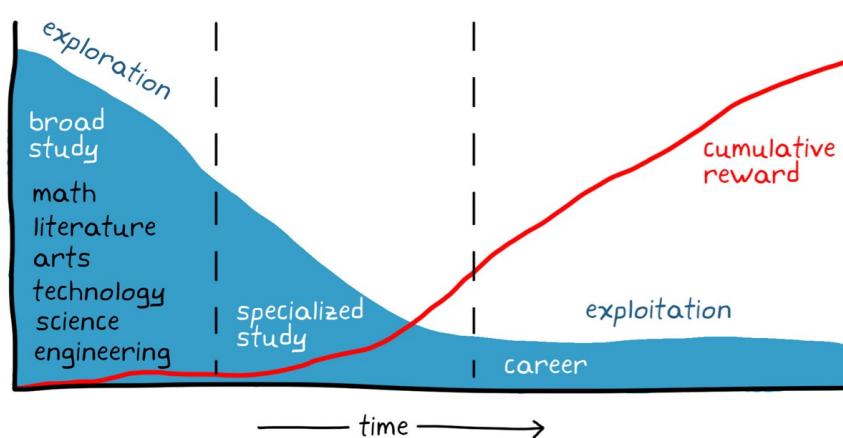
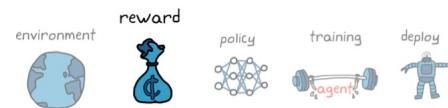


85



85

Balancing Exploration / Exploitation

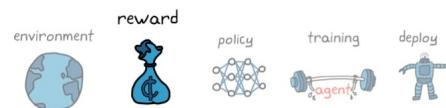


86



86

The Value of Value



reward state

-1	+1	0	-1	+5
S_0	S_1	S_2	S_3	S_4

agent

going left is the better option

value reward state

+1		+4		
-1	+1	0	-1	+5
S_0	S_1	S_2	S_3	S_4

agent

going right has higher value

A second critical aspect of reinforcement learning is the concept of **value**. Assessing the value of a state or an action, rather than reward, helps the agent choose the action that will collect the most rewards over time rather than a short-term benefit.

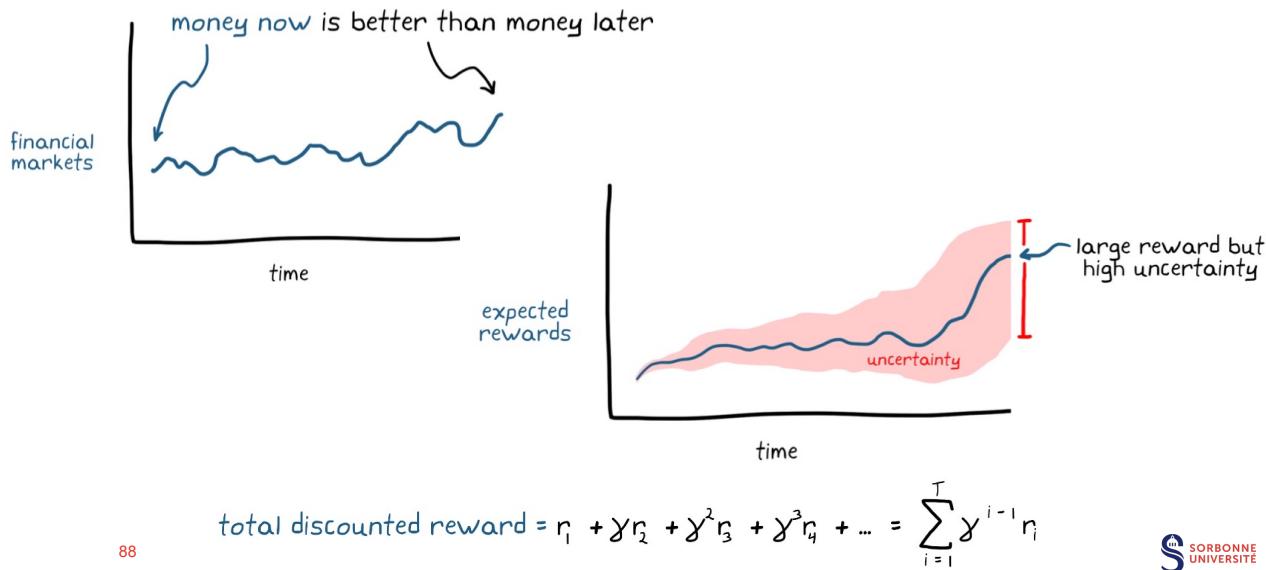
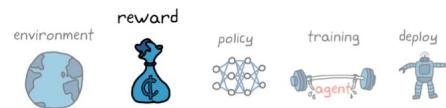
reward: the instantaneous benefit of being in a state and taking a specific action

value: the total rewards an agent expects to receive from a state and onwards into the future



87

L'avantage d'être myope The Benefit of Being Short-Sighted



88



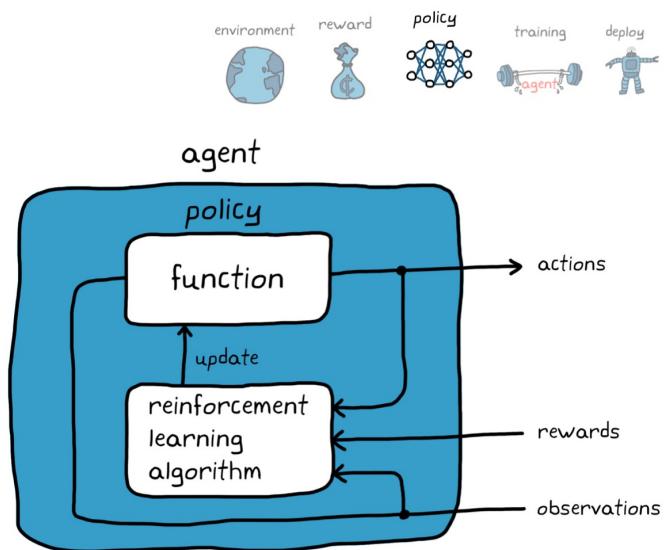
Qu'est la Politique?

Maintenant que vous comprenez l'environnement et son rôle dans la fourniture de l'état et des récompenses, vous êtes prêt à commencer à travailler sur l'agent lui-même.

L'agent est composé de la politique et de l'algorithme d'apprentissage.

La politique est la fonction qui mappe les observations aux actions, et l'algorithme d'apprentissage est la méthode d'optimisation utilisée pour trouver la politique optimale.

89



89

What Is the Policy?

Now that you understand the environment and its role in providing the state and the rewards, you're ready to start work on the agent itself.

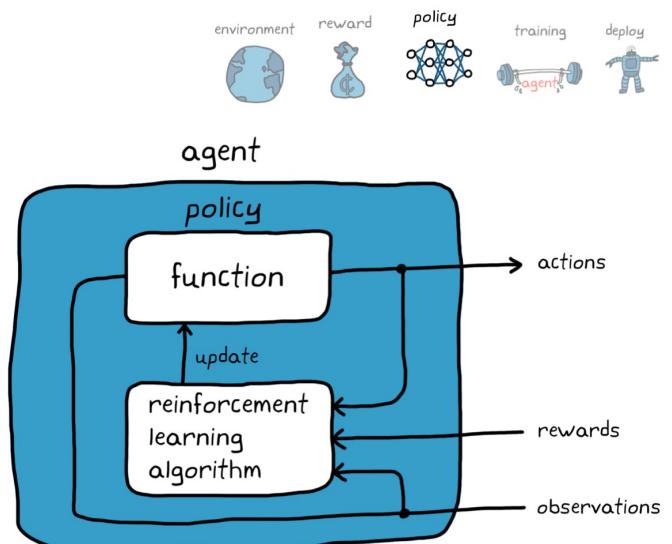
The agent is composed of the policy and the learning algorithm.

The policy is the function that maps observations to actions, and the learning algorithm is the optimization method used to find the optimal policy.

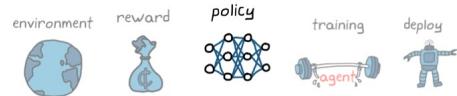
90



90



Représenter une politique



Une politique = une fonction qui prend en compte les observations de l'état et génère des actions.

$$\text{policy} \Rightarrow \text{actions} = \text{function(state observations)}$$

Il existe deux approches pour structurer la fonction politique :

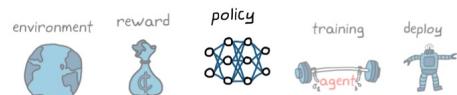
- Direct : Il existe une correspondance spécifique entre les observations de l'état et l'action.
- Indirect : vous examinez d'autres métriques comme la valeur pour déduire le mappage optimal.

91



91

Representing a Policy



A policy = a function that takes in state observations and outputs actions.

$$\text{policy} \Rightarrow \text{actions} = \text{function(state observations)}$$

There are two approaches for structuring the policy function:

- Direct: There is a specific mapping between state observations and action.
- Indirect: You look at other metrics like value to infer the optimal mapping.

92



92

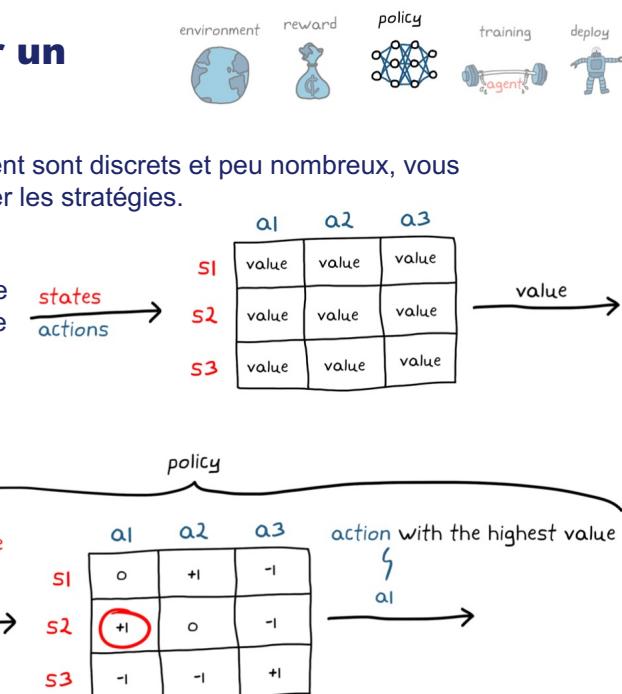
Représenter une stratégie par un tableau

Si les espaces d'état et d'action de l'environnement sont discrets et peu nombreux, vous pouvez utiliser un tableau simple pour représenter les stratégies.

Tableaux de nombres où une entrée agit comme une adresse de recherche et la sortie est le nombre correspondant dans le tableau. (c'est-à-dire Q-table - mappe les états et les actions à la valeur).

Q-table - la politique consiste à vérifier la valeur de chaque action possible compte tenu de l'état actuel, puis à choisir l'action avec la valeur la plus élevée. Former un agent avec une table Q consisterait à déterminer les valeurs correctes pour chaque couple état/action dans la table.

93



93

Representing a Policy with a Table

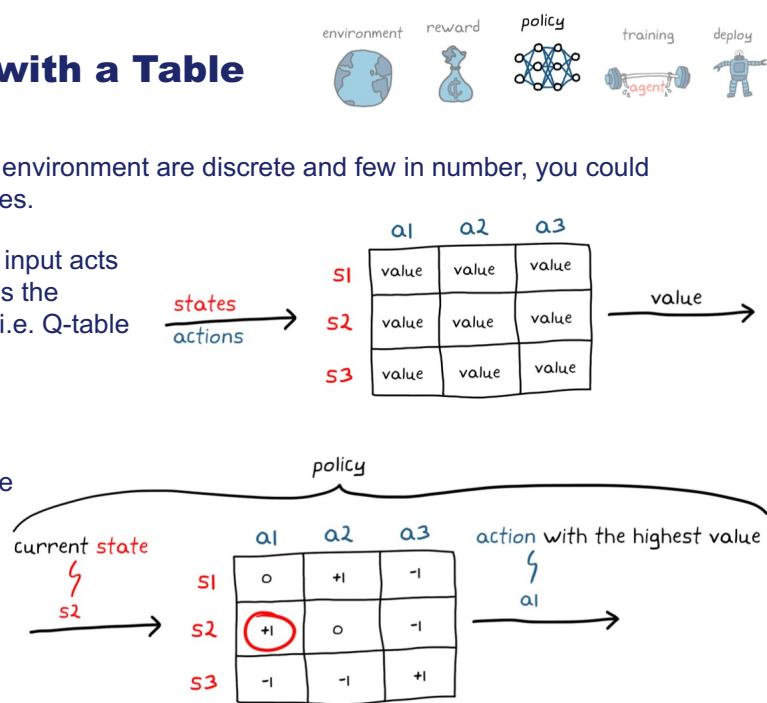
If the state and action spaces for the environment are discrete and few in number, you could use a simple table to represent policies.

Tables = array of numbers where an input acts as a lookup address and the output is the corresponding number in the table. (i.e. Q-table - maps states and actions to value).

Q-table - the policy is to check the value of every possible action given the current state and then choose the action with the highest value.

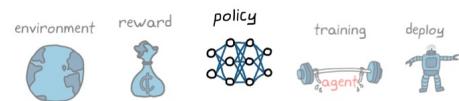
Training an agent with a Q-table would consist of determining the correct values for each state/action pair in the table.

94

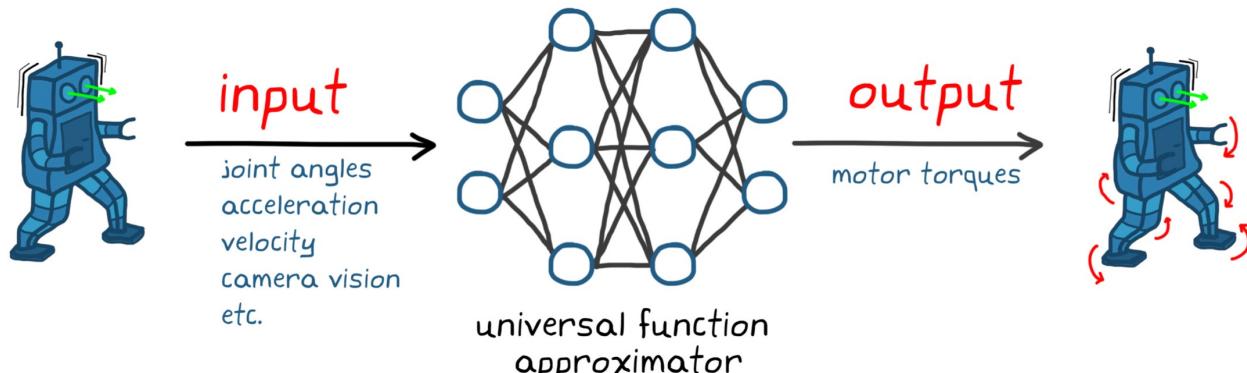


94

Représenter une politique avec un réseau de neurones

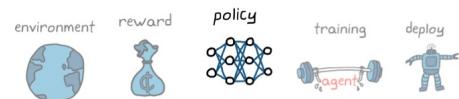


Vous voulez trouver une fonction qui peut prendre un grand nombre d'observations et les transformer en un ensemble d'actions qui contrôleront un environnement non linéaire. Étant donné que la structure de cette fonction est souvent trop complexe pour être résolue directement, vous pouvez (par exemple) l'approximer par un réseau de neurones qui apprend la fonction au fil du temps.

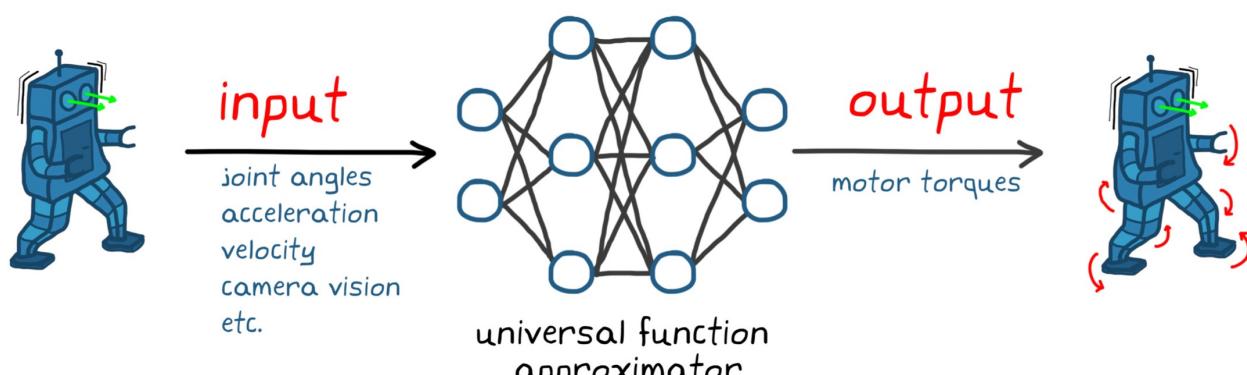


95

Representing a Policy with a Neural Network

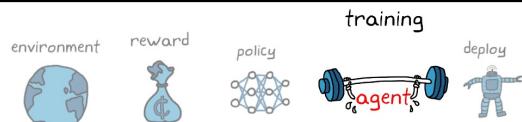


You want to find a function that can take in a large number of observations and transform them into a set of actions that will control some nonlinear environment. Since the structure of this function is often too complex to solve for directly, you can (for example) approximate it with a neural network that learns the function over time.



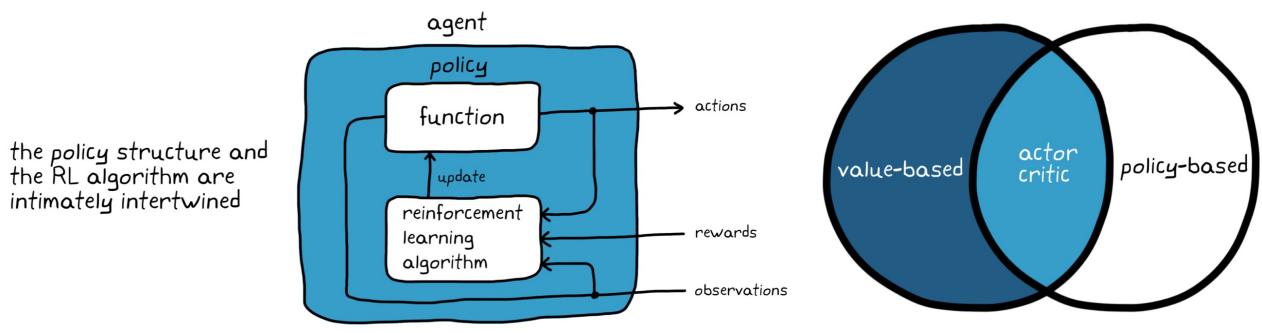
96

Comment la politique est-elle structurée ?



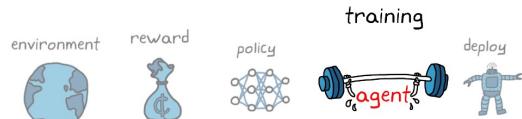
Dans un algorithme d'apprentissage par renforcement (A/R), **la structure politique et l'algorithme d'apprentissage par renforcement sont intimement liés** ; vous ne pouvez pas structurer la politique sans également choisir l'algorithme d'A/R.

Nous décrirons les approches basées sur les fonctions politiques, basées sur les fonctions de valeur et critiques des acteurs pour l'apprentissage par renforcement afin de mettre en évidence les différences dans les structures politiques.



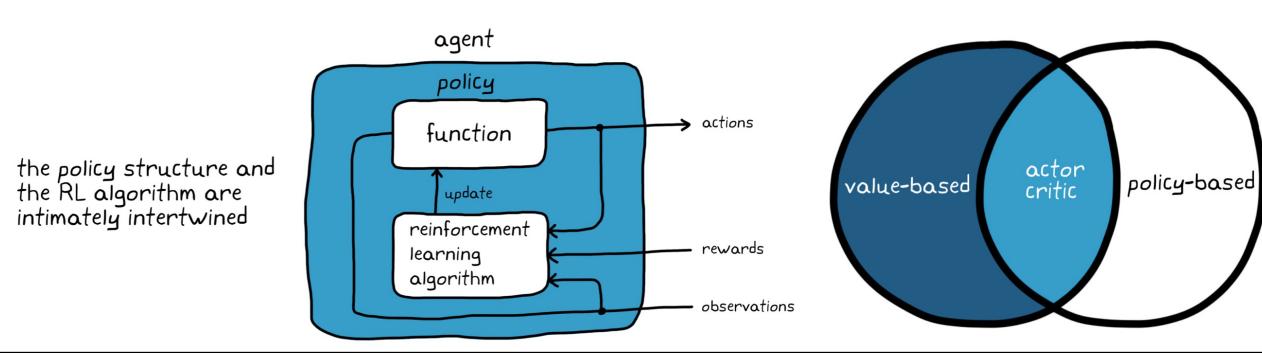
97

How the Policy Is Structured ?



In a reinforcement learning algorithm, the **policy structure and the reinforcement learning algorithm are intimately intertwined**; you can't structure the policy without also choosing the RL algorithm.

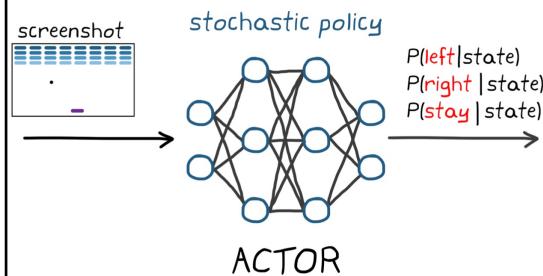
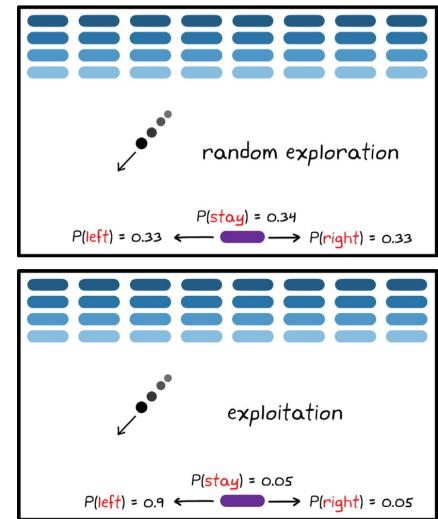
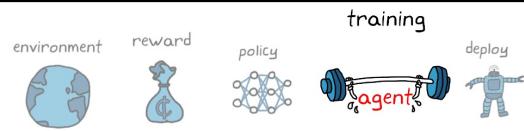
We will describe policy function-based, value function-based, and actor-critic approaches to reinforcement learning to highlight the differences in the policy structures.



98

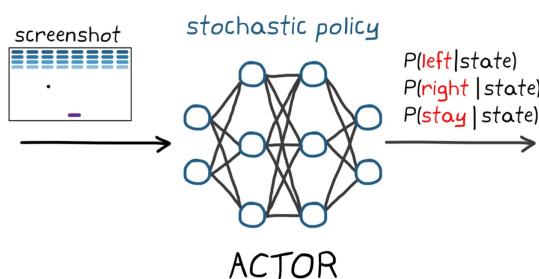
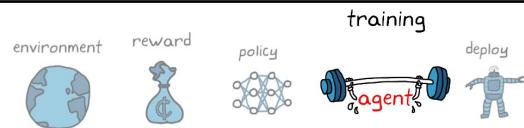
Une politique stochastique d'apprentissage

L'agent doit-il exploiter l'environnement en choisissant les actions qui collectent le plus de récompenses qu'il connaît déjà, ou doit-il choisir des actions qui explorent des parties de l'environnement encore inconnues ? Une politique stochastique aborde ce compromis en intégrant l'exploration dans les probabilités. Désormais, lorsque l'agent apprend, il lui suffit de mettre à jour les probabilités. Prendre à gauche est-il une meilleure option que de prendre à droite ? Si c'est le cas, augmentez la probabilité que vous preniez à gauche dans cet état.

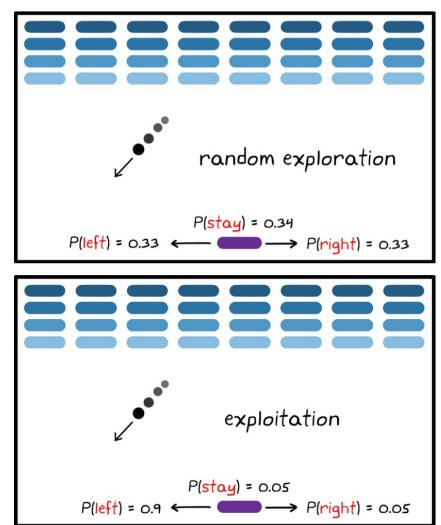


99

A Stochastic Learning Policy



Should the agent exploit the environment by choosing the actions that collect the most rewards that it already knows about, or should it choose actions that explore parts of the environment that are still unknown? A stochastic policy addresses this tradeoff by building exploration into the probabilities. Now, when the agent learns, it just needs to update the probabilities. Is taking a left a better option than taking a right? If so, push the probability that you take a left in this state higher.

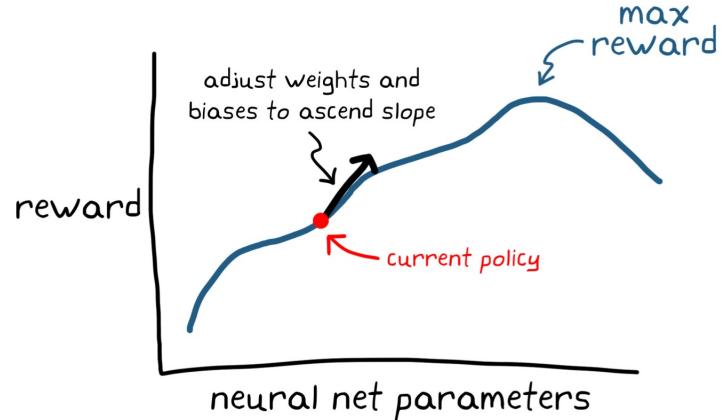
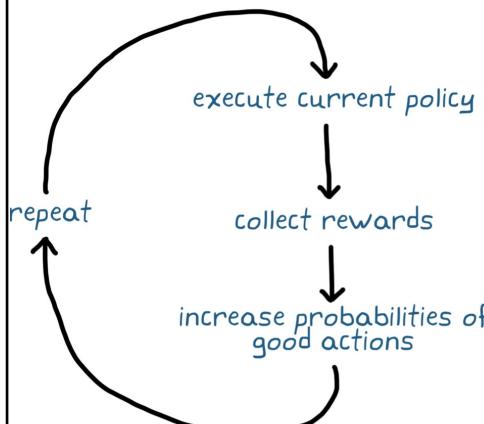
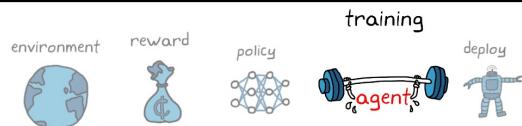


100

100

Stratégie du gradient

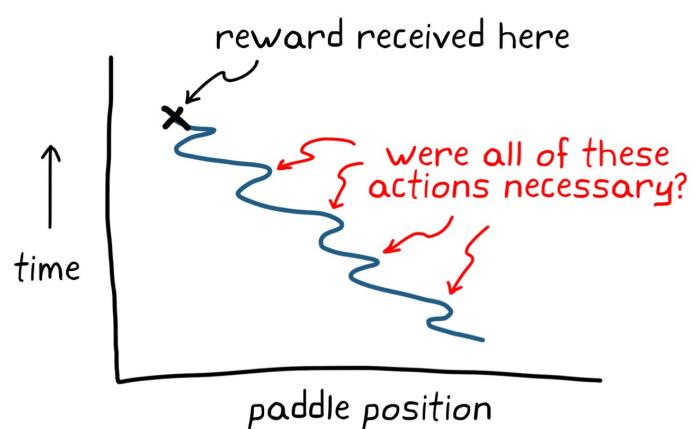
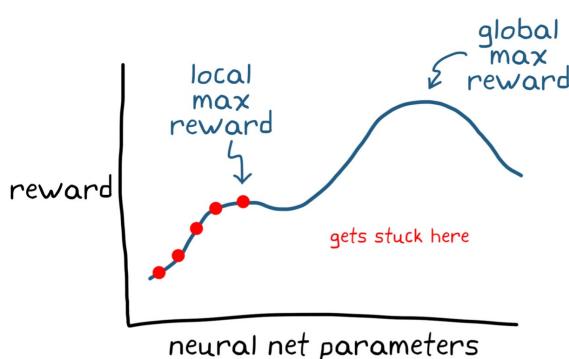
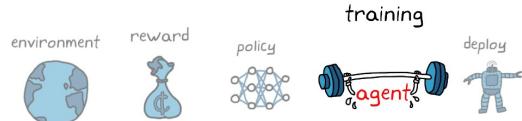
Policy Gradient Methods



101

Inconvénient des stratégies du gradient

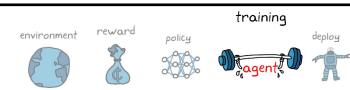
Downside of Policy Gradient Methods



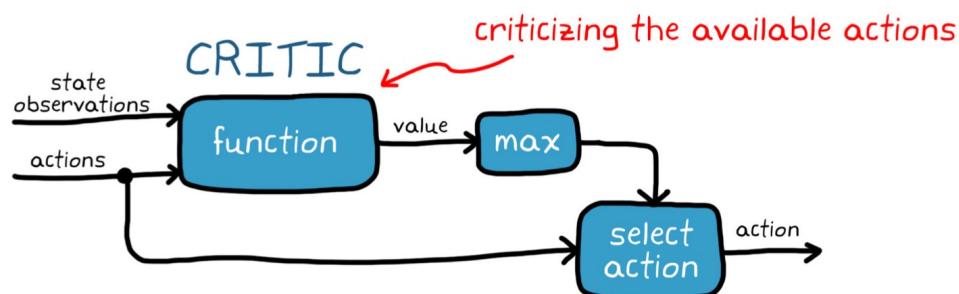
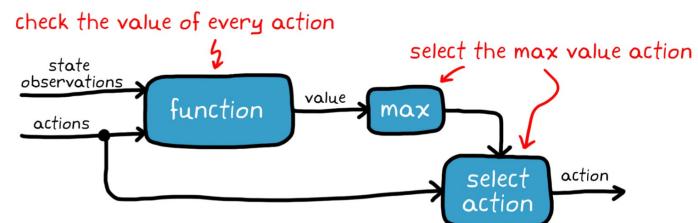
102

Value Function-Based Learning

Apprentissage basé sur la fonction de valeur



what is the current state?
value = function(state observations, action)
how good is the action from this state?



S SORBONNE UNIVERSITÉ

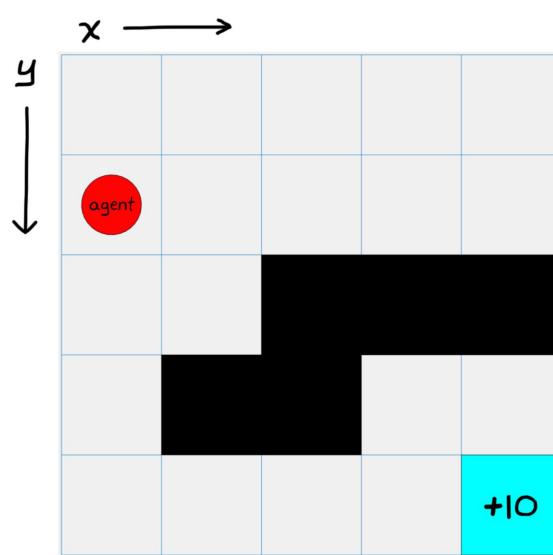
103

Value Functions and Grid World

Fonctions de valeur et Grid World



104

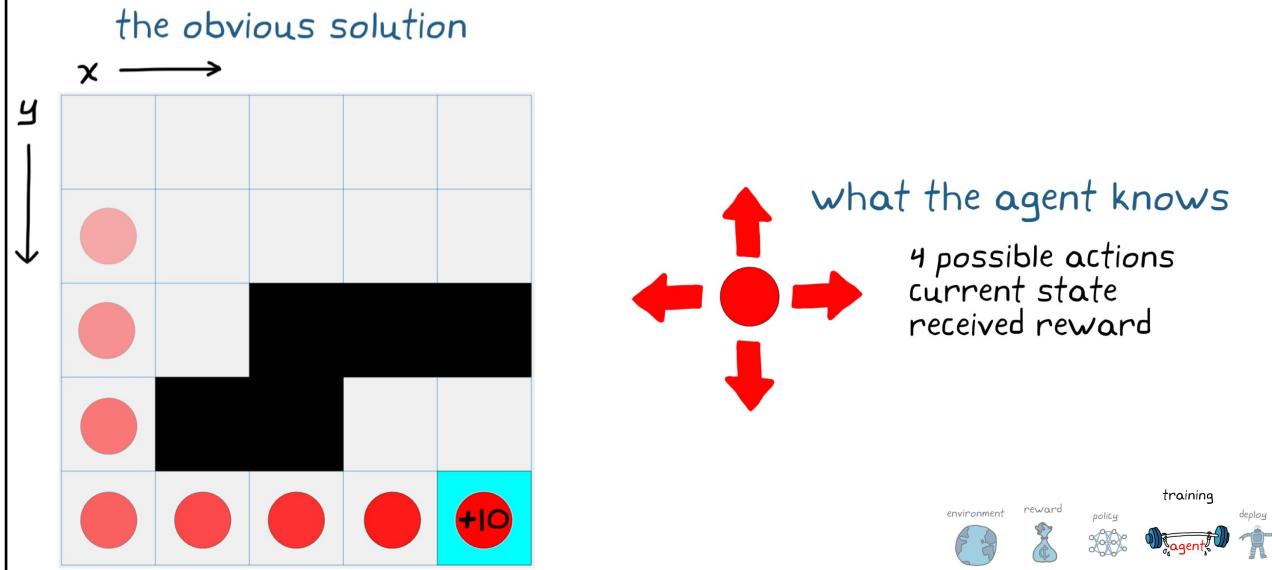


S SORBONNE UNIVERSITÉ

104

Value Functions and Grid World Continued

Fonctions de valeur et Grid World (suite)



105

Résoudre Grid World avec Q-Tables

states → actions →

x	y	up	down	left	right
1	1	value	value	value	value
1	2	value	value	value	value
1	3	value	value	value	value
:					

value →

L'agent acquiert une connaissance de l'environnement en prenant des mesures et en apprenant les valeurs de cette paire état/action en fonction de la récompense reçue. Comme il existe un nombre fini d'états et d'actions dans le monde de la grille, vous pouvez utiliser une table Q pour les mapper à des valeurs. Alors, comment l'agent apprend-il ces valeurs ? Grâce à un processus appelé Q-learning.

Avec Q-learning, vous pouvez commencer par initialiser la table à zéro, afin que toutes les actions aient la même apparence pour l'agent. Une fois que l'agent a effectué une action aléatoire, il passe à un nouvel état et récupère la récompense de l'environnement. L'agent utilise cette récompense comme nouvelle information pour mettre à jour la valeur de l'état précédent et l'action qu'il vient d'entreprendre en utilisant la célèbre équation de Bellman.

initial state

states → actions →

x	y	up	down	left	right
1	1	o	o	o	o
1	2	o	o	o	o
1	3	o	o	o	o
:					

value →

106

Solving Grid World with Q-Tables



x	y	up	down	left	right
1	1	value	value	value	value
1	2	value	value	value	value
1	3	value	value	value	value
:					

states → actions → value →

With Q-learning, you can start by initializing the table to zeros, so all actions look the same to the agent. After the agent takes a random action, it gets to a new state and collects the reward from the environment. The agent uses that reward as new information to update the value of the previous state and the action that it just took using the famous Bellman equation.

The way the agent builds up knowledge of the environment is by taking actions and learning the values of that state/action pair based on the received reward. Since there are a finite number of states and actions in grid world, you can use a Q-table to map them to values. So how does the agent learn these values? Through a process called Q-learning.

initial state

x	y	up	down	left	right
1	1	o	o	o	o
1	2	o	o	o	o
1	3	o	o	o	o
:					

states → actions → value →

107

The Bellman Equation



$$\text{new } Q(s, a) = Q(s, a) + \alpha [R(s, a) + \gamma \cdot \max Q'(s', a') - Q(s, a)]$$

$$\text{new } Q(s, a) = Q(s, a) + \alpha [R(s, a) + \gamma \cdot \max Q'(s', a') - Q(s, a)]$$

↑
reward for taking action, a, from state, s

$$\text{new } Q(s, a) = Q(s, a) + \alpha [R(s, a) + \gamma \cdot \underline{\max Q'(s', a')} - Q(s, a)]$$

↑
maximum expected value from state, s'

$$\text{new } Q(s, a) = Q(s, a) + \alpha [R(s, a) + \gamma \cdot \underline{\max Q'(s', a')} - Q(s, a)]$$

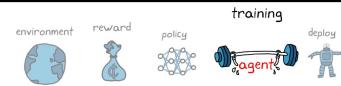
↑
discount future rewards

108



108

The Bellman Equation *Continued*



Q-table has been updated

$$\text{new } Q(s, a) = Q(s, a) + \alpha \left[R(s, a) + \gamma \max Q'(s', a') - \underbrace{Q(s, a)}_{\substack{\text{new best estimate of value} \\ \downarrow \text{previous estimate of value}}} \right]$$

$$\text{new } Q(s, a) = Q(s, a) + \alpha \left[R(s, a) + \gamma \max Q'(s', a') - \underbrace{Q(s, a)}_{\substack{\uparrow \text{error is multiplied by learning rate}}} \right]$$

$$\text{new } Q(s, a) = \underbrace{Q(s, a)}_{\substack{\uparrow \text{delta value is added to old estimate}}} + \alpha \left[R(s, a) + \gamma \max Q'(s', a') - Q(s, a) \right]$$

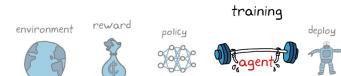
The Bellman equation is another connection between reinforcement learning and traditional control theory. If you are familiar with optimal control theory, you may notice that this equation is the discrete version of the Hamilton-Jacobi-Bellman equation, which, when solved over the entire state space, is a necessary and sufficient condition for an optimum.

109



109

L'équation de Bellman (suite)



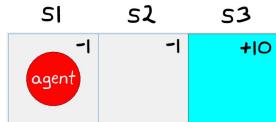
S1	S2	S3
-1 agent	-1	+10
agent		

Il peut être utile de voir l'équation de Bellman en action en examinant les premières étapes d'un exemple simple de Grid World : alpha est défini sur 1 et gamma est défini sur 0,9. Si les deux actions ont la même valeur, alors l'agent effectue une action aléatoire ; sinon, l'agent choisit l'action avec la valeur la plus élevée.

Episode	Step	State	current Q(s, a)	Action	R(s, a)	new Q(s, a)																								
1	1	S1	<table border="1"> <thead> <tr> <th></th> <th>S1</th> <th>S2</th> <th>S3</th> </tr> </thead> <tbody> <tr> <td>Left</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>Right</td> <td>0</td> <td>0</td> <td>0</td> </tr> </tbody> </table>		S1	S2	S3	Left	0	0	0	Right	0	0	0	right (random)	-1	<table border="1"> <thead> <tr> <th></th> <th>S1</th> <th>S2</th> <th>S3</th> </tr> </thead> <tbody> <tr> <td>Left</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>Right</td> <td>-1</td> <td>0</td> <td>0</td> </tr> </tbody> </table>		S1	S2	S3	Left	0	0	0	Right	-1	0	0
	S1	S2	S3																											
Left	0	0	0																											
Right	0	0	0																											
	S1	S2	S3																											
Left	0	0	0																											
Right	-1	0	0																											
					Bellman equation: $0 + 1[-1 + 0.9 \cdot 0 - 0] = -1$																									
1	2	S2	<table border="1"> <thead> <tr> <th></th> <th>S1</th> <th>S2</th> <th>S3</th> </tr> </thead> <tbody> <tr> <td>Left</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>Right</td> <td>0</td> <td>0</td> <td>0</td> </tr> </tbody> </table>		S1	S2	S3	Left	0	0	0	Right	0	0	0	right (random)	+10	<table border="1"> <thead> <tr> <th></th> <th>S1</th> <th>S2</th> <th>S3</th> </tr> </thead> <tbody> <tr> <td>Left</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>Right</td> <td>-1</td> <td>10</td> <td>0</td> </tr> </tbody> </table>		S1	S2	S3	Left	0	0	0	Right	-1	10	0
	S1	S2	S3																											
Left	0	0	0																											
Right	0	0	0																											
	S1	S2	S3																											
Left	0	0	0																											
Right	-1	10	0																											
					Bellman equation: $0 + 1[10 + 0.9 \cdot 0 - 0] = +10$																									
End of episode																														
110																														
Lorsque l'agent atteint l'état de fin, S3, l'épisode se termine et l'agent se réinitialise à l'état de départ, S1. Les valeurs de la table Q persistent et l'apprentissage se poursuit dans l'épisode suivant, qui se poursuit sur la page suivante.																														

110

The Bellman Equation Continued



It might be helpful to see the Bellman equation in action by looking at the first few steps in a simple Grid World example : alpha is set to 1 and gamma is set to 0.9. If both actions have the same value, then the agent takes a random action; otherwise, the agent chooses the action with the highest value.

Episode	Step	State	current Q(s, a)	Action	R(s, a)	new Q(s, a)																								
1	1	S1	<table border="1"> <thead> <tr> <th></th> <th>S1</th> <th>S2</th> <th>S3</th> </tr> </thead> <tbody> <tr> <td>Left</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>Right</td> <td>0</td> <td>0</td> <td>0</td> </tr> </tbody> </table>		S1	S2	S3	Left	0	0	0	Right	0	0	0	right (random)	-1	<table border="1"> <thead> <tr> <th></th> <th>S1</th> <th>S2</th> <th>S3</th> </tr> </thead> <tbody> <tr> <td>Left</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>Right</td> <td>-1</td> <td>0</td> <td>0</td> </tr> </tbody> </table>		S1	S2	S3	Left	0	0	0	Right	-1	0	0
	S1	S2	S3																											
Left	0	0	0																											
Right	0	0	0																											
	S1	S2	S3																											
Left	0	0	0																											
Right	-1	0	0																											
1	2	S2	<table border="1"> <thead> <tr> <th></th> <th>S1</th> <th>S2</th> <th>S3</th> </tr> </thead> <tbody> <tr> <td>Left</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>Right</td> <td>0</td> <td>0</td> <td>0</td> </tr> </tbody> </table>		S1	S2	S3	Left	0	0	0	Right	0	0	0	right (random)	+10	<table border="1"> <thead> <tr> <th></th> <th>S1</th> <th>S2</th> <th>S3</th> </tr> </thead> <tbody> <tr> <td>Left</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>Right</td> <td>-1</td> <td>10</td> <td>0</td> </tr> </tbody> </table>		S1	S2	S3	Left	0	0	0	Right	-1	10	0
	S1	S2	S3																											
Left	0	0	0																											
Right	0	0	0																											
	S1	S2	S3																											
Left	0	0	0																											
Right	-1	10	0																											
End of episode																														
<p>When the agent reaches the termination state, S3, the episode ends and the agent reinitializes at the starting state, S1. The Q-table values persist and the learning continues into the next episode, which is continued on the next page.</p>																														



111

L'équation de Bellman (suite)



Episode	Step	State	current Q(s, a)	Action	R(s, a)	new Q(s, a)																								
2	1	S1	<table border="1"> <thead> <tr> <th></th> <th>S1</th> <th>S2</th> <th>S3</th> </tr> </thead> <tbody> <tr> <td>Left</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>Right</td> <td>-1</td> <td>10</td> <td>0</td> </tr> </tbody> </table>		S1	S2	S3	Left	0	0	0	Right	-1	10	0	left (greedy)	-1	<table border="1"> <thead> <tr> <th></th> <th>S1</th> <th>S2</th> <th>S3</th> </tr> </thead> <tbody> <tr> <td>Left</td> <td>-1</td> <td>0</td> <td>0</td> </tr> <tr> <td>Right</td> <td>-1</td> <td>10</td> <td>0</td> </tr> </tbody> </table>		S1	S2	S3	Left	-1	0	0	Right	-1	10	0
	S1	S2	S3																											
Left	0	0	0																											
Right	-1	10	0																											
	S1	S2	S3																											
Left	-1	0	0																											
Right	-1	10	0																											
2	2	S1	<table border="1"> <thead> <tr> <th></th> <th>S1</th> <th>S2</th> <th>S3</th> </tr> </thead> <tbody> <tr> <td>Left</td> <td>-1</td> <td>0</td> <td>0</td> </tr> <tr> <td>Right</td> <td>-1</td> <td>10</td> <td>0</td> </tr> </tbody> </table>		S1	S2	S3	Left	-1	0	0	Right	-1	10	0	right (random)	-1	<table border="1"> <thead> <tr> <th></th> <th>S1</th> <th>S2</th> <th>S3</th> </tr> </thead> <tbody> <tr> <td>Left</td> <td>-1</td> <td>0</td> <td>0</td> </tr> <tr> <td>Right</td> <td>8</td> <td>10</td> <td>0</td> </tr> </tbody> </table>		S1	S2	S3	Left	-1	0	0	Right	8	10	0
	S1	S2	S3																											
Left	-1	0	0																											
Right	-1	10	0																											
	S1	S2	S3																											
Left	-1	0	0																											
Right	8	10	0																											
End of episode																														

En seulement quatre actions, l'agent s'est déjà installé sur une table Q qui produit la politique optimale ; dans l'état S1, il prendra à droite puisque la valeur 8 est supérieure à -1, et dans l'état S2, il prendra à nouveau à droite puisque la valeur 10 est supérieure à 0. Ce qui est intéressant dans ce résultat, c'est que la Q-table ne s'est pas arrêté sur les vraies valeurs de chaque paire état/action. S'il continue à apprendre, les valeurs continueront à se déplacer dans le sens des valeurs réelles. Cependant, vous n'avez pas besoin de trouver les vraies valeurs pour produire la politique optimale ; vous avez juste besoin que la valeur de l'action optimale soit le nombre le plus élevé.

112

The Bellman Equation *Continued*



Episode	Step	State	current Q(s, a)	Action	R(s, a)	new Q(s, a)																								
2	1	S1	<table border="1"> <thead> <tr> <th></th><th>S1</th><th>S2</th><th>S3</th></tr> </thead> <tbody> <tr> <td>Left</td><td>0</td><td>0</td><td>0</td></tr> <tr> <td>Right</td><td>-1</td><td>10</td><td>0</td></tr> </tbody> </table>		S1	S2	S3	Left	0	0	0	Right	-1	10	0	left (greedy)	-1	<table border="1"> <thead> <tr> <th></th><th>S1</th><th>S2</th><th>S3</th></tr> </thead> <tbody> <tr> <td>Left</td><td>-1</td><td>0</td><td>0</td></tr> <tr> <td>Right</td><td>-1 ↑</td><td>10</td><td>0</td></tr> </tbody> </table> <p>Bellman equation: $0 + 1 \cdot [-1 + 0.9 \cdot 0 - (-1)] = -1$</p>		S1	S2	S3	Left	-1	0	0	Right	-1 ↑	10	0
	S1	S2	S3																											
Left	0	0	0																											
Right	-1	10	0																											
	S1	S2	S3																											
Left	-1	0	0																											
Right	-1 ↑	10	0																											
2	2	S1	<table border="1"> <thead> <tr> <th></th><th>S1</th><th>S2</th><th>S3</th></tr> </thead> <tbody> <tr> <td>Left</td><td>-1</td><td>0</td><td>0</td></tr> <tr> <td>Right</td><td>-1</td><td>10</td><td>0</td></tr> </tbody> </table>		S1	S2	S3	Left	-1	0	0	Right	-1	10	0	right (random)	-1	<table border="1"> <thead> <tr> <th></th><th>S1</th><th>S2</th><th>S3</th></tr> </thead> <tbody> <tr> <td>Left</td><td>-1</td><td>0</td><td>0</td></tr> <tr> <td>Right</td><td>8</td><td>10</td><td>0</td></tr> </tbody> </table> <p>Bellman equation: $-1 + 1 \cdot [-1 + 0.9 \cdot 10 - (-1)] = +8$</p>		S1	S2	S3	Left	-1	0	0	Right	8	10	0
	S1	S2	S3																											
Left	-1	0	0																											
Right	-1	10	0																											
	S1	S2	S3																											
Left	-1	0	0																											
Right	8	10	0																											

End of episode

Within just four actions, the agent has already settled on a Q-table that produces the optimal policy; in state S1, it will take a right since the value 8 is higher than -1, and in state S2, it will take a right again since the value 10 is higher than 0. What's interesting about this result is that the Q-table hasn't settled on the true values of each state/action pair. If it keeps learning, the values will continue to move in the direction of the actual values. However, you don't need to find the true values in order to produce the optimal policy; you just need the value of the optimal action to be the highest number.



113

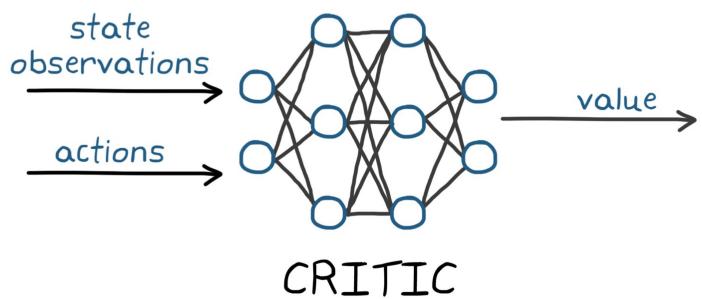
The Critic as a Neural Network



states are θ and $\dot{\theta}$

Étendez cette idée à un pendule inversé. Comme Grid World, il y a deux états, l'angle et la vitesse angulaire, sauf que maintenant les états sont continus.

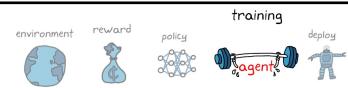
value function-based learning



La fonction valeur (la critique) est représentée par un réseau de neurones. L'idée est la même qu'avec un tableau : vous saisissez les observations d'état et une action, le réseau de neurones renvoie la valeur de cette paire état/action et la politique consiste à choisir l'action avec la valeur la plus élevée. Au fil du temps, le réseau convergerait lentement vers une fonction qui produirait la vraie valeur pour chaque action n'importe où dans l'espace d'état continu.

114

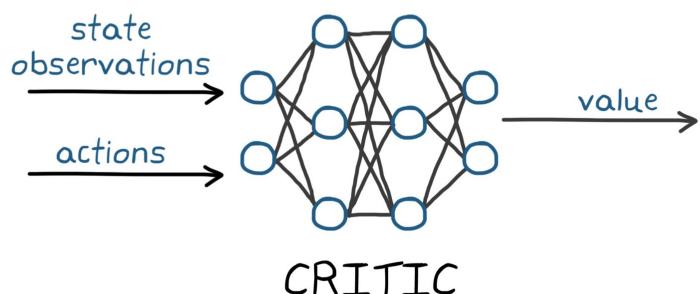
The Critic as a Neural Network



states are θ and $\dot{\theta}$

Extend this idea to an inverted pendulum. Like Grid World, there are two states, angle and angular rate, except now the states are continuous.

value function-based learning



The value function (the critic) is represented with a neural network. The idea is the same as with a table: You input the state observations and an action, the neural network returns the value of that state/action pair, and the policy is to choose the action with the highest value. Over time, the network would slowly converge on a function that outputs the true value for every action anywhere in the continuous state space.

115

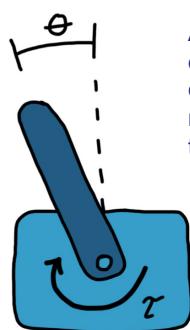
115

Les inconvénients des politiques basées sur la valeur



Vous pouvez utiliser un réseau de neurones pour définir la fonction de valeur pour les espaces d'état continus. Si le pendule inversé a un espace d'action discret, vous pouvez introduire des actions discrètes dans votre réseau critique une par une.

Les stratégies basées sur la fonction de valeur ne fonctionneront pas bien pour les espaces d'action continu. C'est parce qu'il n'y a aucun moyen de calculer la valeur une à la fois pour chaque action infinie pour trouver la valeur maximale. Même pour un espace d'action grand (mais pas infini), cela devient coûteux en termes de calcul. C'est malheureux car souvent, dans les problèmes de contrôle, vous avez un espace d'action continu, comme l'application d'une plage continue de couples à un problème de pendule inversé.



Alors, que peut-on faire? : Implémenter une méthode de gradient de stratégie vanille, comme indiqué dans la section algorithme basé sur la fonction de stratégie. Ces algorithmes peuvent gérer des espaces d'action continus, mais ils ont du mal à converger lorsqu'il y a une grande variance dans les récompenses et que le gradient est bruyant. Alternativement, vous pouvez fusionner les deux techniques d'apprentissage dans une classe d'algorithmes appelée acteur-critique.

continuous states: θ and $\dot{\theta}$

discrete action space: $\mathcal{A} = [-2, -1, 0, 1, 2] \text{ Nm}$



116

The Downside of Value-Based Policies



You can use a neural network to define the value function for continuous state spaces. If the inverted pendulum has a discrete action space, you can feed discrete actions into your critic network one at a time. Value function-based policies won't work well for continuous action spaces. This is because there is no way to calculate the value one at a time for every infinite action to find the maximum value. Even for a large (but not infinite) action space, this becomes computationally expensive. This is unfortunate because often in control problems you have a continuous action space, such as applying a continuous range of torques to an inverted pendulum problem.



So what can you do?

You can implement a [vanilla policy gradient method](#), as covered in the policy function-based algorithm section. These algorithms can handle continuous action spaces, but they have trouble converging when there is high variance in the rewards and the gradient is noisy. Alternatively, you can merge the two learning techniques into a class of algorithms called actor-critic.

continuous states: θ and $\dot{\theta}$

discrete action space: $\mathcal{A} = [-2, -1, 0, 1, 2] \text{ Nm}$

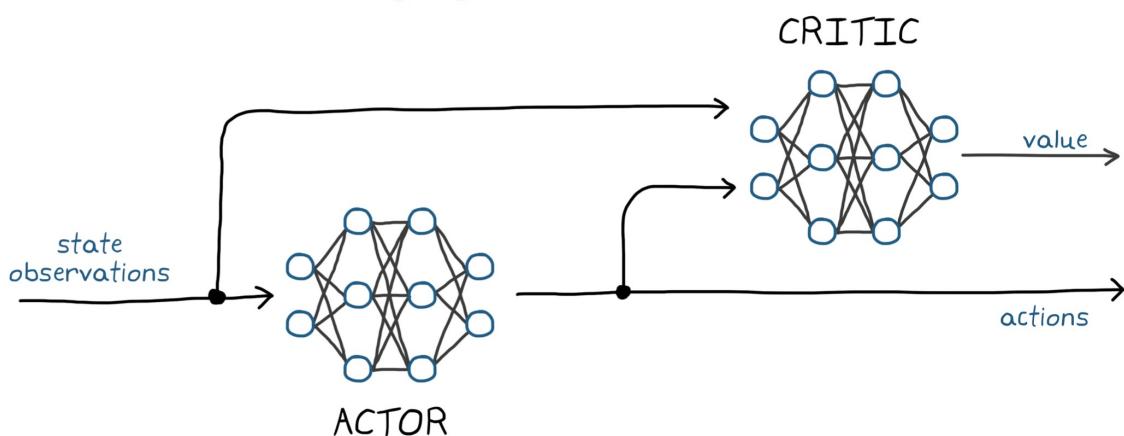


117

Méthodes Acteur-Critique Actor-Critic Methods



actor-critic learning algorithms



118

Le cycle d'apprentissage acteur-critique

The diagram illustrates the Actor-Critic learning cycle. It features two neural network components: an **ACTOR** and a **CRITIC**. The **ACTOR** receives **state observations** and produces **actions**. The **CRITIC** receives both **state observations** and **actions** from the **ACTOR**, and produces a **value**.

Legend:

- environment
- reward
- policy
- training
- agent
- deploy

Text:

L'acteur choisit une action de la même manière qu'un algorithme de fonction de politique, et elle est appliquée à l'environnement.

Le critique fait une prédiction de la valeur de cette action pour l'état actuel et la paire d'action.

119

119

The Actor-Critic Learning Cycle

The diagram illustrates the Actor-Critic learning cycle. It features two neural network components: an **ACTOR** and a **CRITIC**. The **ACTOR** receives **state observations** and produces **actions**. The **CRITIC** receives both **state observations** and **actions** from the **ACTOR**, and produces a **value**.

Legend:

- environment
- reward
- policy
- training
- agent
- deploy

Text:

The actor chooses an action in the same way that a policy function algorithm would, and it is applied to the environment.

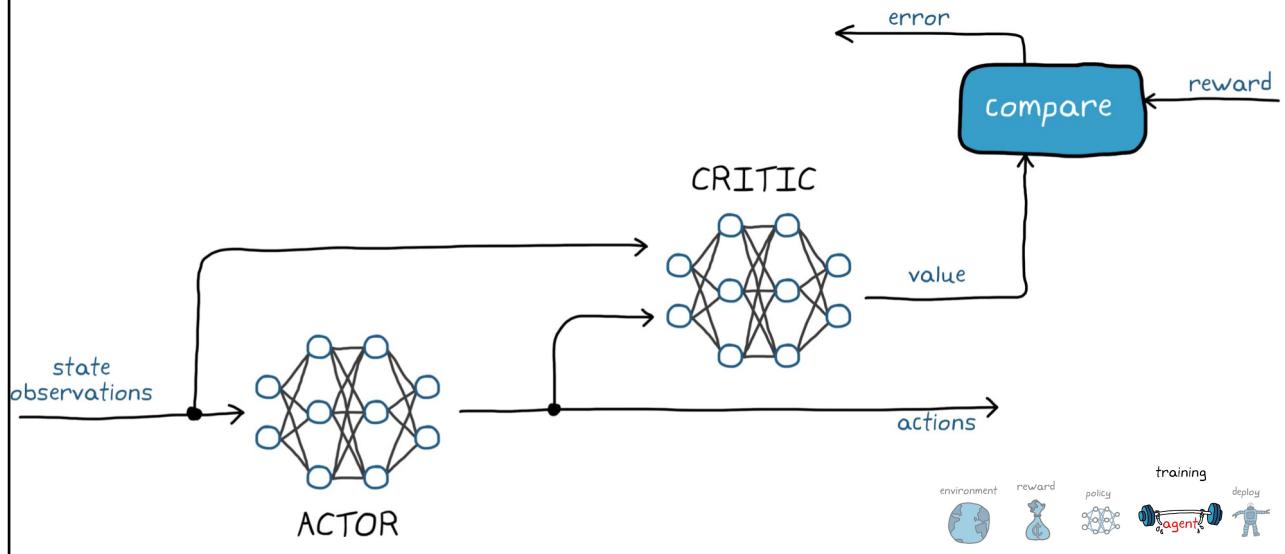
The critic makes a prediction of what the value of that action is for the current state and action pair.

120

120

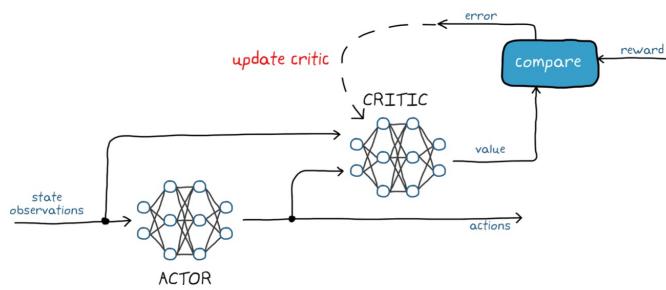
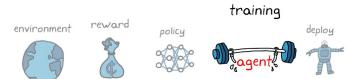
Le cycle d'apprentissage acteur-critique

The Actor-Critic Learning Cycle Continued



121

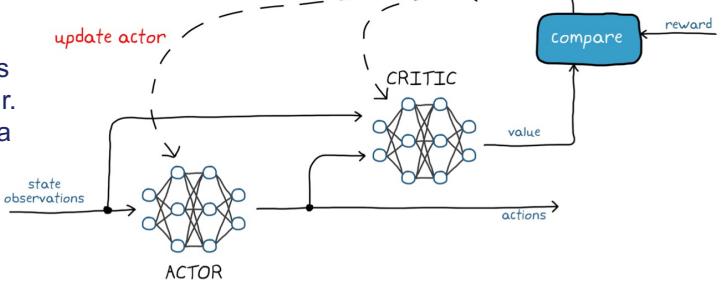
Le cycle d'apprentissage acteur-critique



L'acteur se met également à jour avec la réponse du critique afin qu'il puisse ajuster ses probabilités de reprendre cette action à l'avenir. De cette façon, la politique monte désormais la pente des récompenses dans la direction recommandée par le critique plutôt que d'utiliser les récompenses directement.

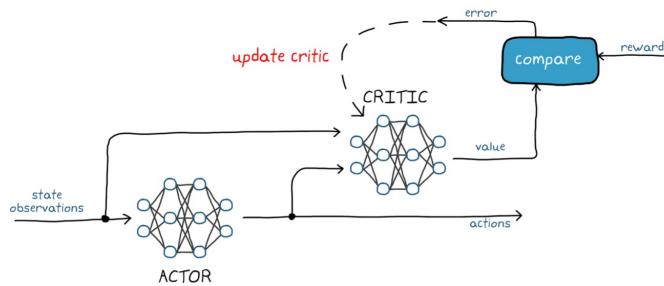
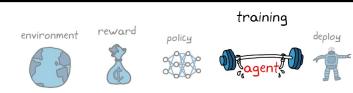
122

Le critique utilise cette erreur pour se mettre à jour de la même manière qu'une fonction de valeur le ferait afin d'avoir une meilleure prédiction la prochaine fois qu'il sera dans cet état.



122

The Actor-Critic Learning Cycle

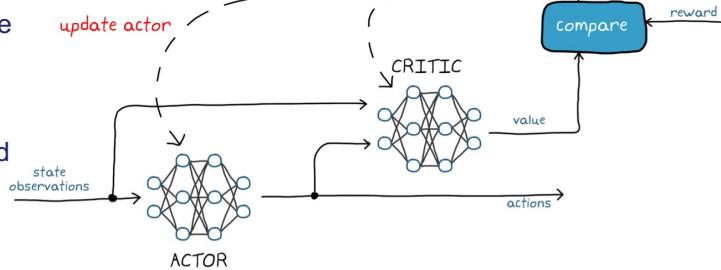


The critic uses this error to update itself in the same way that a value function would so that it has a better prediction the next time it's in this state.

The actor also updates itself with the response from the critic so that it can adjust its probabilities of taking that action again in the future.

In this way, the policy now ascends the reward slope in the direction that the critic recommends rather than using the rewards directly.

123

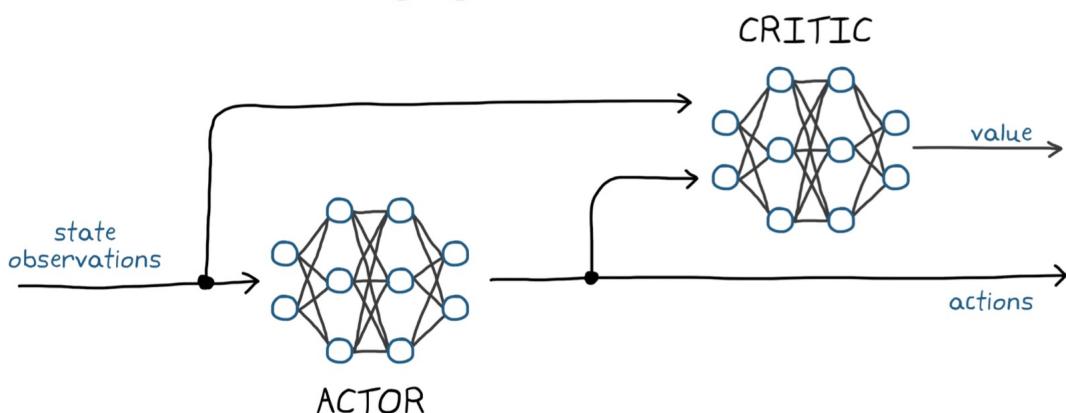


123

Deux réseaux complémentaires Two Complementing Networks



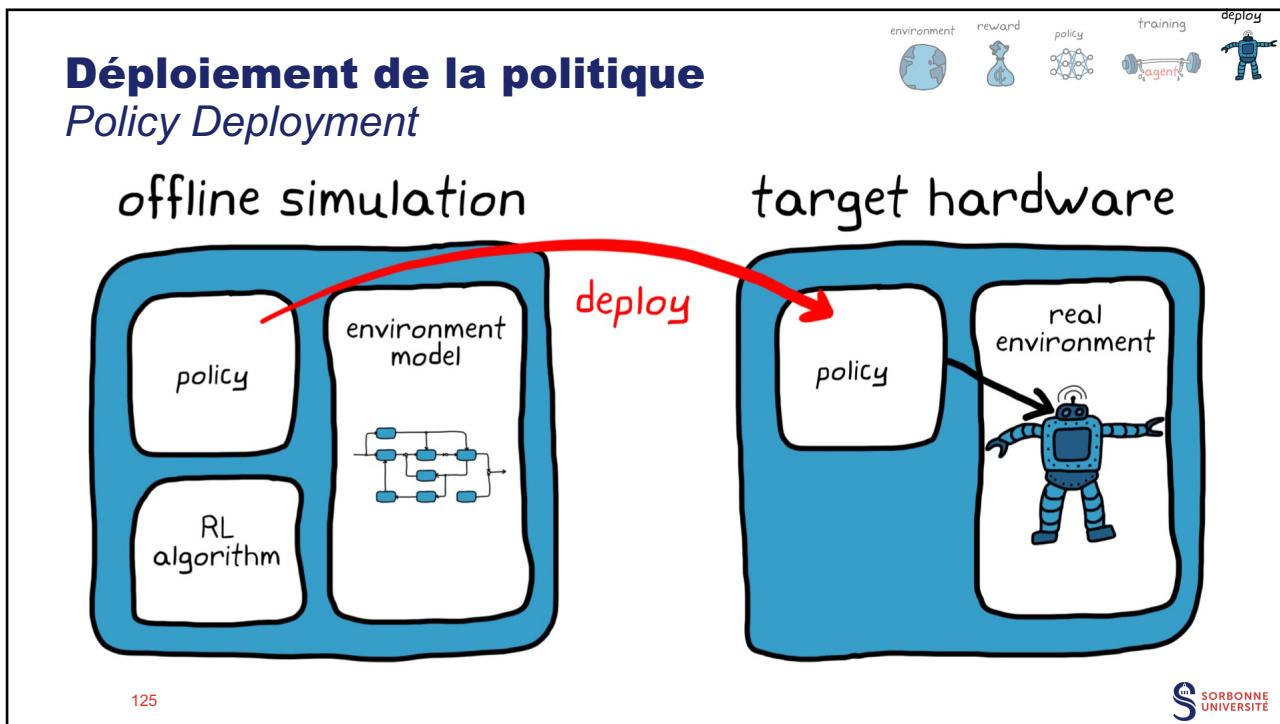
actor / critic learning algorithms



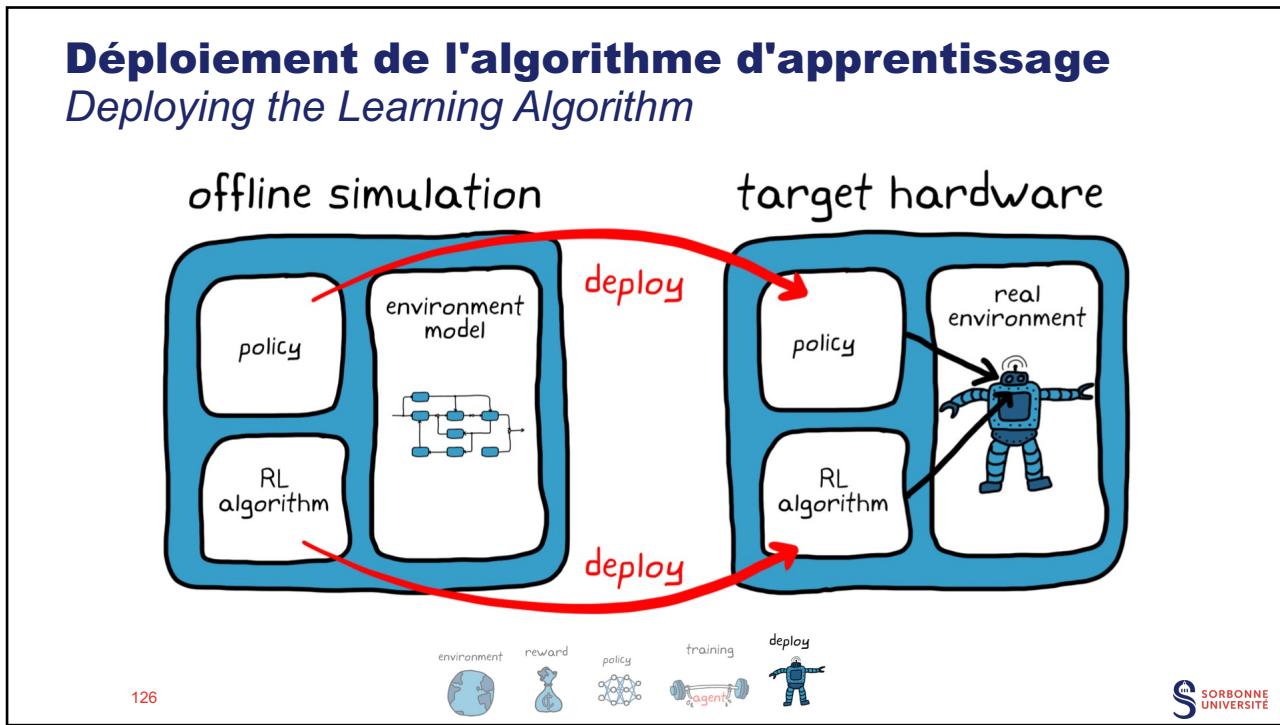
124



124



125



126

La relation complémentaire The Complementary Relationship

start learning here

environment reward policy training agent deploy

coarse-tuned optimal policy

finish learning here

fine-tuned optimal policy

127

127

Les inconvénients de A/R The Drawbacks of RL

agent

environment

actions → reward ← observations

resulting policy

state observations → actions

PERFECT

MEH

These challenges come down to two main questions:

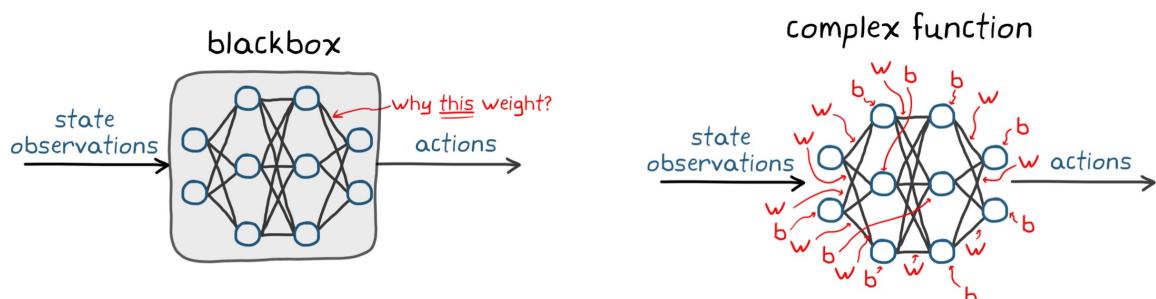
1. How do you know the solution is going to work?
2. Is there a way to manually adjust it if it's not quite perfect?

128

128

Le réseau de neurones inexplicable

The Unexplainable Neural Network



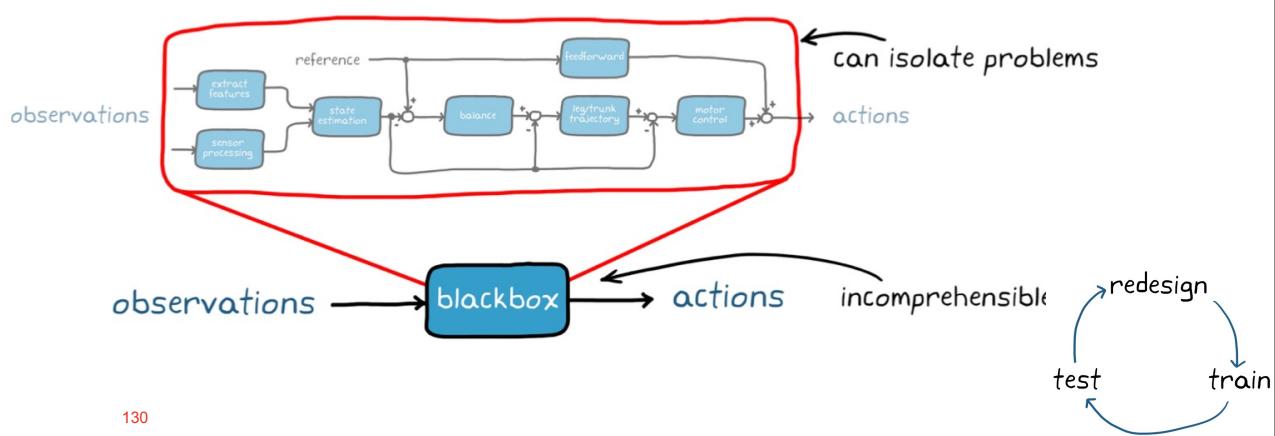
129



129

Identification des problèmes

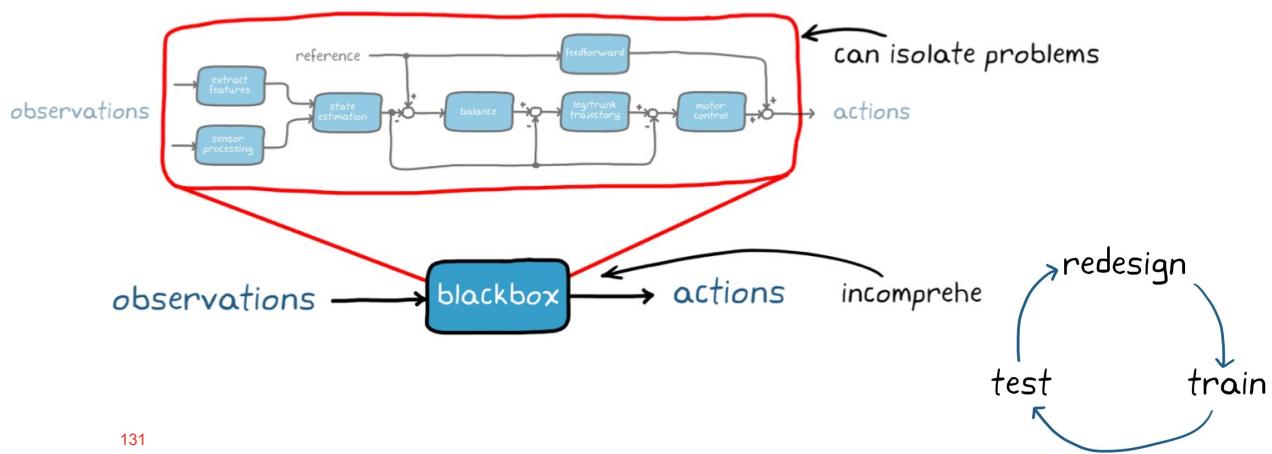
Il y a un problème où ce qui a rendu la résolution du problème de contrôle plus facile – condenser la logique difficile à une seule fonction de boîte noire – a rendu notre solution finale, **incompréhensible**.



130

Pinpointing Problems

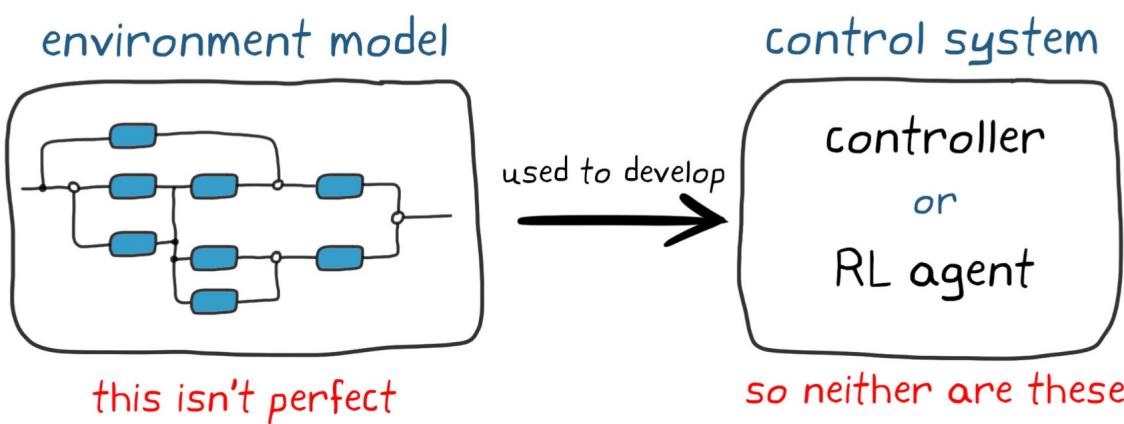
There is an issue where the very thing that has made solving the control problem easier—condensing the difficult logic down to a single black-box function—has made our final solution **incomprehensible**.



131

Le plus gros problème

Il y a un problème plus important qui se profile ici qui va au-delà du temps nécessaire pour former un agent, et il se résume à la précision du modèle d'environnement.

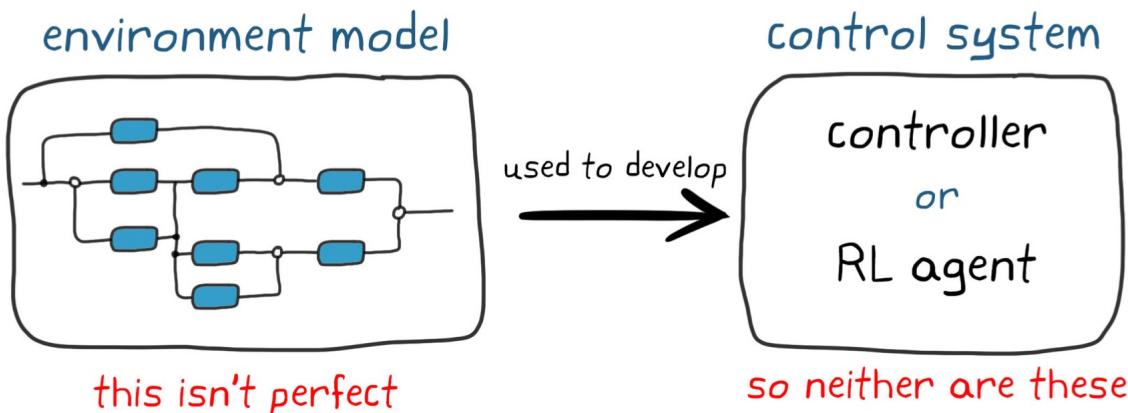


132

132

The Larger Problem

There is a larger issue looming here that goes beyond the length of time it takes to train an agent, and it comes down to the accuracy of the environment model.

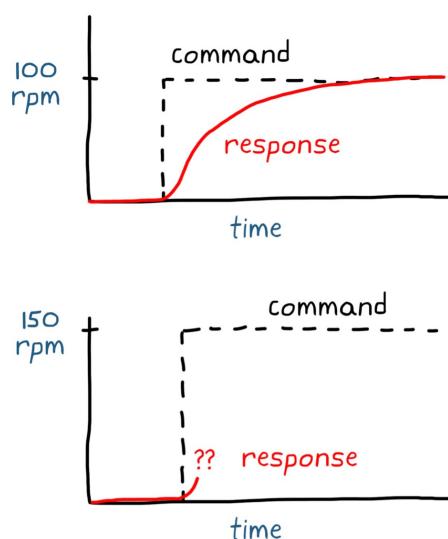


133



133

Vérification de la politique apprise Verifying the Learned Policy



134



134

Vérification de la politique apprise

Formal Verification Method

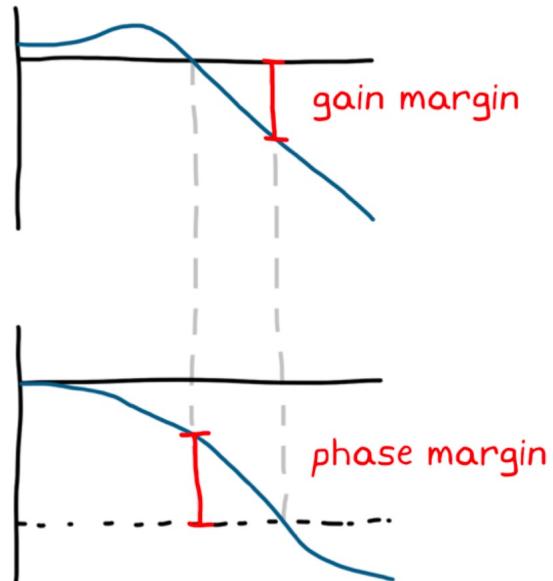
Les réseaux de neurones appris rendent également la vérification formelle difficile. Ces méthodes consistent à garantir qu'une condition sera remplie en fournissant une preuve formelle plutôt qu'en utilisant un test.

$$x = \text{abs}(y)$$

verify this is positive

code inspection shows this is true

Learned neural networks also make formal verification difficult. These methods involve guaranteeing that some condition will be met by providing a formal proof rather than using a test.

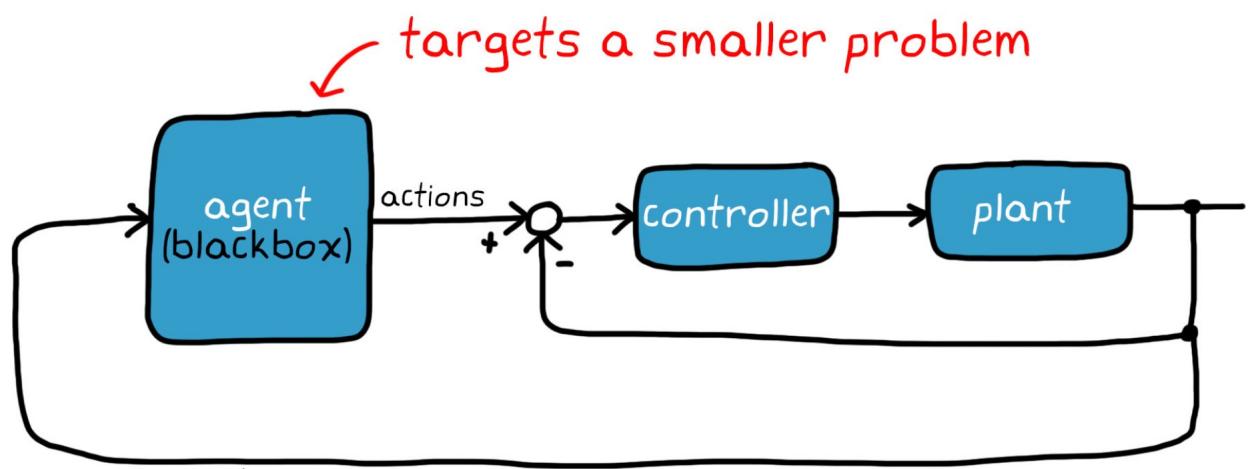


135

135

Réduire le problème

Shrinking the Problem



136

136

Travailler sur ces problèmes

Même si vous ne pouvez pas quantifier la robustesse, la stabilité et la sécurité, vous pouvez résoudre ces problèmes avec des solutions de contournement dans la conception.

episode	torque	length	delay	reference	...
1	2 Nm	1 cm	10 ms	step	.
2	2.5 Nm	1.3 cm	8 ms	ramp	.
3	2.1 Nm	1.7 cm	14 ms	impulse	.
.
.
.

% software monitor

```
if abs(body_angle) > 45; % monitor for falling
    mode = "safe"; % set safe mode
    extend_arms(); % prepare for impact
end
```

137



137

Working Around These Issues

Even though you can't quantify robustness, stability, and safety, you can address those issues with workarounds in the design.

episode	torque	length	delay	reference	...
1	2 Nm	1 cm	10 ms	step	.
2	2.5 Nm	1.3 cm	8 ms	ramp	.
3	2.1 Nm	1.7 cm	14 ms	impulse	.
.
.
.

% software monitor

```
if abs(body_angle) > 45; % monitor for falling
    mode = "safe"; % set safe mode
    extend_arms(); % prepare for impact
end
```

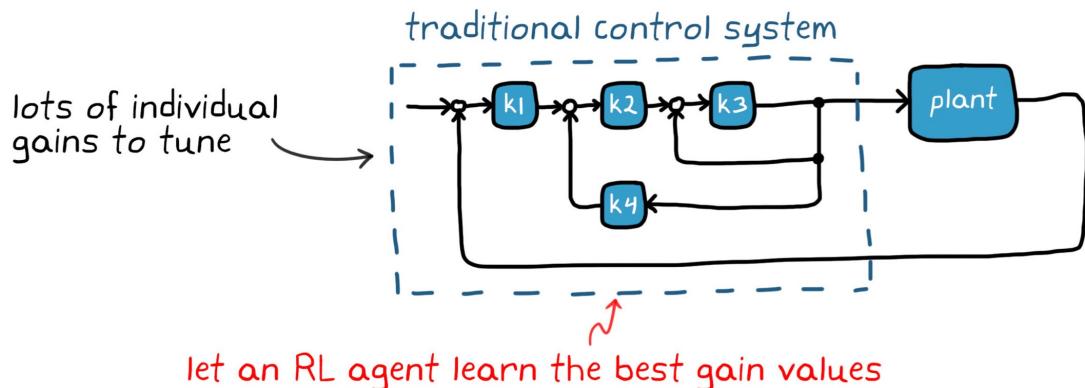
138



138

Résoudre un problème différent

Solving a Different Problem



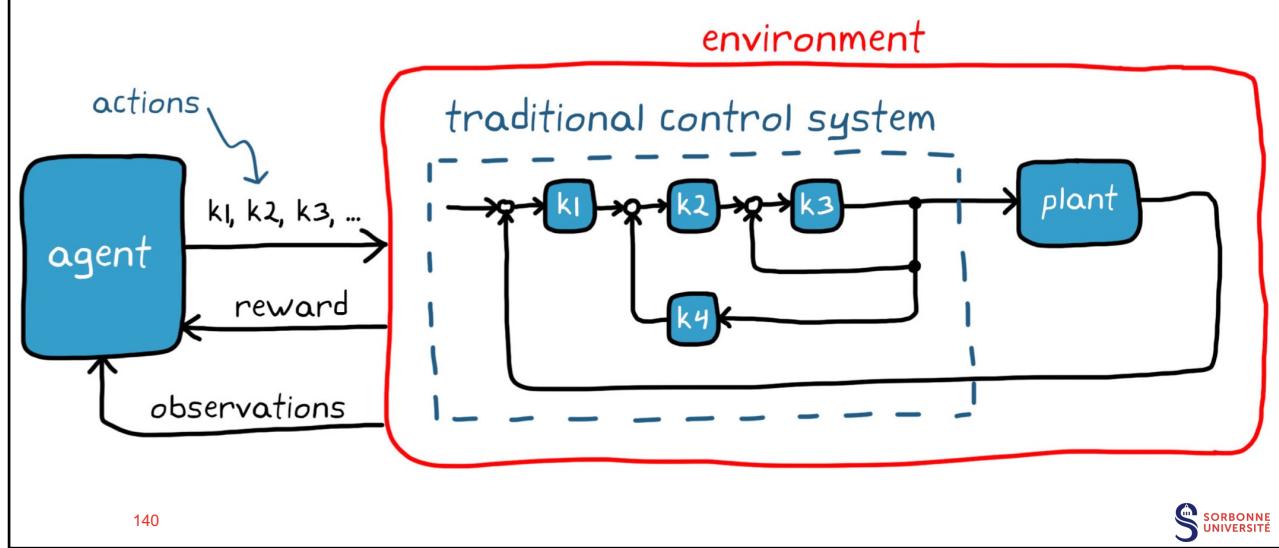
139



139

A/R complétant les méthodes traditionnelles

RL Complementing Traditional Methods



140



140

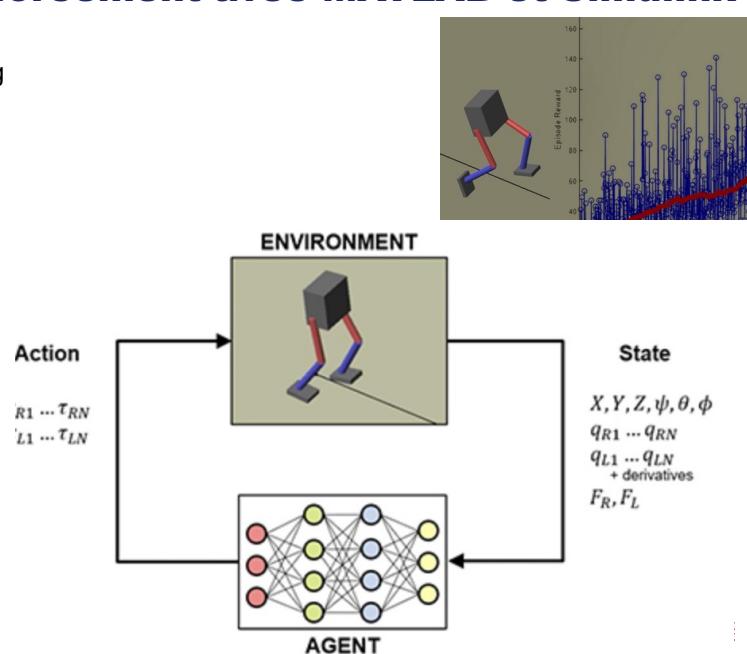
Apprentissage par renforcement avec MATLAB et Simulink

La boîte à outils « Reinforcement Learning Toolbox » fournit des fonctions et des blocs pour les politiques de formation à l'aide d'algorithmes d'apprentissage par renforcement.

Utilisez ces politiques pour implémenter des contrôleurs et des algorithmes de prise de décision pour des systèmes complexes tels que des robots et des systèmes autonomes.

La boîte à outils vous permet de former des politiques en leur permettant d'interagir avec des environnements représentés dans MATLAB ou à l'aide de modèles Simulink.

141



141

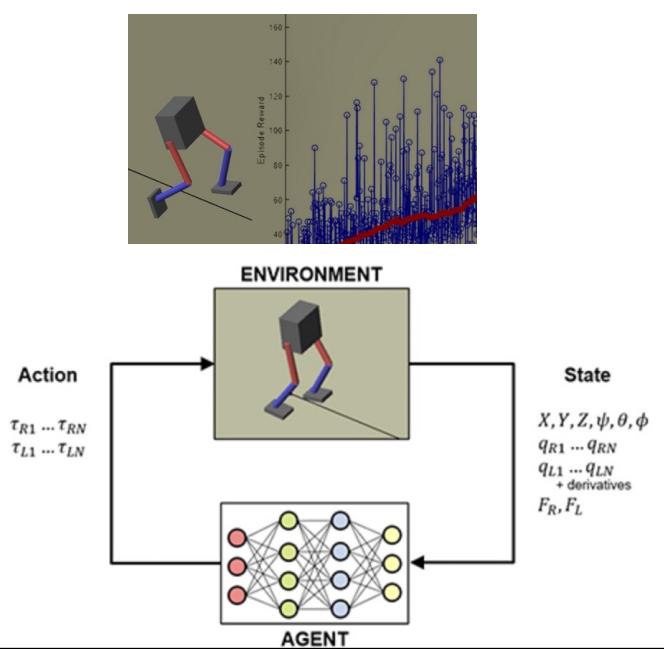
Reinforcement Learning with MATLAB and Simulink

Reinforcement Learning Toolbox provides functions and blocks for training policies using reinforcement learning algorithms.

Use these policies to implement controllers and decision-making algorithms for complex systems such as robots and autonomous systems.

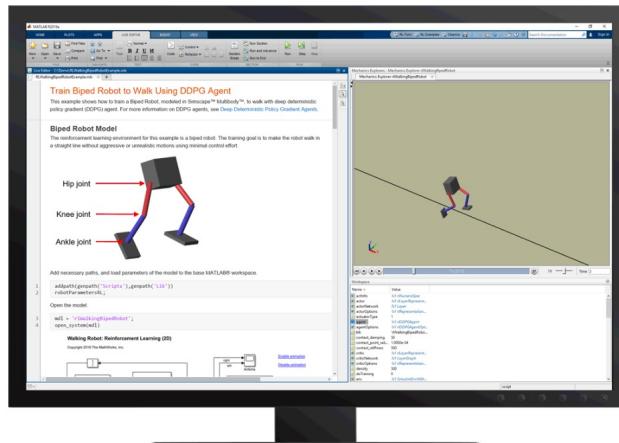
The toolbox lets you train policies by enabling them to interact with environments represented in MATLAB or using Simulink models.

142

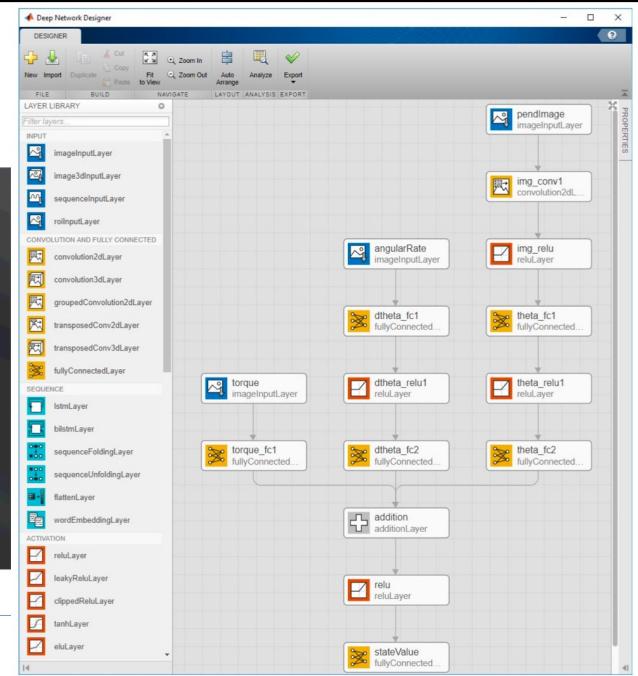


142

Reinforcement Learning with MATLAB and Simulink



143



Deep Q-learning network (DQN) agent created with the Deep Network Designer app.

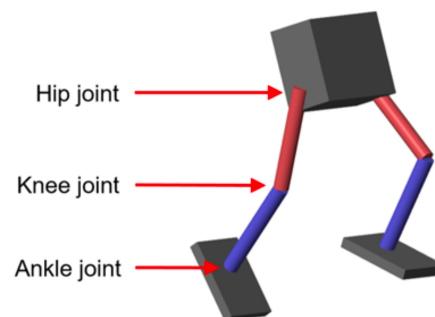
143

The Future of Reinforcement Learning

Reinforcement learning is a powerful tool for solving hard problems. There

here are some challenges regarding understanding the solution and verifying that it will work, but as we covered, you have a few ways right now to work around those challenges.

While reinforcement learning is nowhere near its full potential, it may not be too long before it becomes the design method of choice for all complex control systems.



144



144

AlphaGo : quand l'intelligence artificielle dépasse l'homme

Système intelligent développé par DeepMind(Google).

Capable de jouer d'une façon autonome le jeu chinois GO.

Utilise une méthode **d'apprentissage par renforcement** (Monte Carlo) mixée avec **l'apprentissage profond** (deep learning).

En Mai 2017 ,AlphaGo a gagné le champion Ke Jie.

145



145

AlphaGo : quand l'intelligence artificielle dépasse l'homme



Figure : AlphaGO écrase Ke Jie

146



146

Quand l'intelligence artificielle dépasse l'homme

Au début(alpha zero), ne connaît rien au jeu de Go (from scratch), sauf les règles.
Ses premiers coups sont joués aléatoirement.

En fonction du résultat de la partie, AlphaGO s'améliorent légèrement.

Voici quelques échelles de temps des progressions de AlphaGO :

- 19 heures : elle maîtrise des concepts stratégiques avancés comme la notion de territoire, de vie et de mort, etc.
- 3 jours : elle arrive au niveau d'un très bon joueur.
- 21 jours : AlphaGo Zero arrive au niveau d'AlphaGo Master, l'IA qui avait terrassé le champion Ke Jie en 2017.
- 40 jours (29 millions de parties) : l'IA dépasse toutes les versions d'AlphaGo et devient virtuellement **le meilleur joueur de Go au monde**.

147



147

Quand l'intelligence artificielle dépasse l'homme



148



148